

FUNCTIONS LISTED BY CATEGORY

Please select the category of functions you would like to view:

Data Information Category

Data Information functions provide information on the specific characteristics of the data to which an analysis technique is applied. For example, say you developed a system that bought on a limit order at the Close plus 5 points or 10 points depending on the data to which the system was applied. On the Yen, you would want to use the 5 points, but on the Deutschemark you would want to use the 10 points. To define this goal, you could use the CommodityNumber function, which returns a unique number for each data series.

Throughout the following functions, Data1 is the default if a data series is not specified. Data2 - Data50 can also be used. Data series are set in the Omega Server or Omega Downloader. Please refer to their respective manuals for more information on defining data series.

The function for which you want a detailed explanation:

BarInterval

Returns the period in minutes of the bar. This is the bar interval selected in the Charting program.

Category:

Data Information

Function:

BarInterval

Example:

Assuming Data1 is IBM 15-minute bars and Data2 is INDU 30-minute bars then:

BarInterval of Data1 returns 15

BarInterval of Data2 returns 30

BigPointValue

Returns the dollar amount for each full point.

Category:

Data Information

Function:

BigPointValue

Example:

To convert dollars to points, use the following:

$\text{Value1} = (\text{TradeProfit}(1) / \text{BigPointValue}) ;$

To convert points to dollars, use the following:

$\text{Value2} = ((\text{EntryPrice}(0) - \text{Close}[0]) * \text{BigPointValue}) ;$

BoxSize

Returns the box size present in the **Point & Figure Compression** option in the Format **Price Data Settings** dialog.

Category:

Data Information

Function:

BoxSize

CommodityNumber

Returns the commodity number of the data contract which is currently being tested. No input values are required. Each commodity has its own unique commodity number. Refer to the Master list for the appropriate number for each commodity.

Category:

Data Information

Function:

CommodityNumber

Example:

Assuming the current data being tested is Gold on the COMEX, the function returns a value of 30

CurrentBar

Returns the bar number of the current bar, the first bar after bar specified by the **Maximum number of bars referenced by a study** setting (known as MaxBarsBack) is bar 1. The function is updated after each bar. Also see BarNumber.

Category:

Data Information

Function:

CurrentBar

DailyLimit

Returns the daily limit (maximum move permitted by the exchange in any one day) in decimal format of the contract being tested. No input values are required. If commodity has no daily limit, 1,000,000 is returned. For example, in wheat, where the maximum daily limit equals 20 cents, 20.0 is returned.

Category:

Data Information

Function:

DailyLimit

Example:

Daily Limit returns 2.00 when the current data being tested is Pork Bellies, where the maximum daily limit equals 200 points

DailyLimit returns 3.00 when the current data being tested is T-Bonds, which trades in 32nds and has a minimum daily limit equal to 96/32nds

DataCompression

Returns a number representing the data compression of the selected data series. Be sure that the setting for the **Maximum number of bars referenced by a study** (known as MaxBarsBack) of your analysis technique is less than the total number of bars in your chart window.

0 = TickBar

1 = Intra-day

2 = Daily

3 = Weekly

4 = Monthly

5 = Point & Figure

Category:

Data Information

Function:

DataCompression

Example:

Assuming the data series in question is an intra-day data series, Value1 returns a 1.

Value1 = DataCompression

DataInUnion

This function is reserved for future use.

Category:

Data Information

Function:

DataInUnion

DeliveryMonth

Returns the delivery month (expiration month) of the data contract which is currently being tested. No input values are required.

Category:

Data Information

Function:

DeliveryMonth

Example:

Assuming the current data being tested is Pork Bellies 07/91, the function returns a value of 7

DeliveryYear

Returns the delivery year (expiration year) of the futures contract which is currently being tested. No input values are required.

Category:

Data Information

Function:

DeliveryYear

Example:

Assuming the current data being tested is Pork Bellies 07/91, the function will return a value of 1991

LastCalcJDate

The LastCalcJDate returns the Julian date of the last bar.

Category:

Data Information

Function:

LastCalcJDate

LastCalcMMTime

The LastCalcMMTime function returns the time of the last bar in minutes since midnight.

Category:

Data Information

Function:

LastCalcMMTime

Example:

If the last bar was completed at 10:00am, the function would return the value of 600.

MaxBarsBack

Returns the **Maximum number of bars referenced by a study** setting (also known as MaxBarsBack). MaxBarsBack refers to the number of bars of data necessary to calculate the rules in a study, system, or function. Your charting application begins calculation on the bar after the number of bars specified by the MaxBarsBack setting.

For example, in order to correctly calculate a 14-day RSI, MaxBarsBack must be set to at least 14. You should set MaxBarsBack to the maximum number required by any rule in your study.

This function is used in repeat and while loops to limit the number of iterations or passes through the loop.

Category:

Data Information

Function:

MaxBarsBack

MinMove

Returns the minimum movement associated with the selected data series.

Category:

Data Information

Function:

MinMove

Example:

In the following example Value1 returns the minimum movement associated with the second data series. Value1 = MinMove of data2

PointValue

Returns the dollar value of a full point move of the contract which is currently being tested. No input values are required.

If the commodity being tested is grains, a full point equals 1/8th; for bonds it equals 1/32nd, for S&P it equals 1/100th.

Category:

Data Information

Function:

PointValue

Example:

Assuming the current data being tested is Pork Bellies, which moves in 100th, a value of 4 is returned.

PriceScale

Returns the PriceScale associated with the selected data series. In the following example Value1 returns the price scale associated with the second data series. If the price scale is 1/100, Value1 will equal 100.

Category:

Data Information

Function:

PriceScale

Example:

Value1 = PriceScale of data2

RevSize

Returns the reversal size present in the Point & Figure Compression Option in the Format Price Data Settings dialog.

Category:

Data Information

Function:

RevSize

Sess1EndTime

Returns the time, in 24-hour military format, that session1 of the indicated market closes. Most markets only have 1 session. Several markets, for example, Treasury Bonds, have 2 sessions. Sess1 is the day session and Sess2 is the evening session. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

Sess1EndTime

Sess1FirstBarTime

Returns the bar time of the first bar of the first session. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

Sess1FirstBarTime

Sess1StartTime

Returns the time, in 24-hour military format, that session1 of the indicated market opens. Most markets only have 1 session; several markets, bonds for example, have 2 sessions. Sess1 is the day session and Sess2 is the evening session. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

Sess1StartTime

Sess2EndTime

Returns the time, in 24-hour military format, that session2 of the indicated market closes. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

Sess2EndTime

Sess2FirstBarTime

Returns the time, in 24-hour military format, of the first bar of the second session. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

Sess2FirstBarTime

Sess2StartTime

Returns the time, in 24-hour military format, that session 2 of the indicated market opens. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

Sess2StartTime

UnionSess1EndTime

When a chart contains more than one data file, returns ending time of last bar for all files of the first session. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

UnionSess1EndTime

Example:

UnionSess1EndTime returns 4:15.

UnionSess1FirstBar

Returns union bar time of the first bar in session 1. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

UnionSess1FirstBar

Example:

UnionSess1FirstBar returns 9:20.

UnionSess1StartTime

Returns union starting time (earliest opening) of the first session. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

UnionSess1StartTime

Example:

UnionSess1StartTime returns 8:20.

UnionSess2EndTime

Returns union ending time of the second session. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

UnionSess2EndTime

UnionSess2FirstBar

Returns union bar time of the first bar in session 2. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

UnionSess2FirstBar

UnionSess2StartTime

Returns union starting time of the second session. This function is used almost exclusively with intra-day data.

Category:

Data Information

Function:

UnionSess2StartTime

Date & Time Category

Date & Time functions are critical in most Studies or Systems because it can be very important to know the time and date on which an event occurred; Date & Time functions enable you to determine this. Each Date & Time function is listed next. The listing includes a description of the function and the appropriate syntax.

The function for which you want a detailed explanation:

BlockNumber

Returns the Security Block Number. Allows you to write a system that can only run with your security block attached to computer to protect your system. This function was left in for compatibility with other Omega products.

Category:

Date & Time

Function:

BlockNumber

Example:

BlockNumber returns 125

CurrentDate

Returns today's calendar date.

Category:

Date & Time

Function:

CurrentDate

Example:

If the date is October 14, 1992 CurrentDate returns 921014

Note: The CurrentDate and CurrentTime functions will display their values in the following way:

1. When working online: At the moment that the study is applied, the functions will give the current date and time, respectively, for whatever data was already plotted in the chart. The date and time will be the date and time of the chart, not the datafeed time (i.e., if your chart is offset 2 hours from the datafeed time, the date and time given by these functions will be offset by 2 hours). Also, any new quote that comes in after the study was applied will also return the date and time of that quote in terms of the chart, not the datafeed time.
2. When working offline: The date and time will be taken from the computers internal clock.

CurrentTime

Returns current up to the minute time. If you are collecting real-time data from a data feed, the time is based on the data feed time. If you are not collecting real-time data from a data feed, the time is based on your computer's system clock.

Category:

Date & Time

Function:

CurrentTime

Example:

If it is currently 2:10pm, CurrentTime returns 1410

Note: The CurrentDate and CurrentTime functions will display their values in the following way:

1. When working online: At the moment that the study is applied, the functions will give the current date and time, respectively, for whatever data was already plotted in the chart. The date and time will be the date and time of the chart, not the datafeed time (i.e., if your chart

is offset 2 hours from the datafeed time, the date and time given by these functions will be offset by 2 hours). Also, any new quote that comes in after the study was applied will also return the date and time of that quote in terms of the chart, not the datafeed time.
2. When working offline: The date and time will be taken from the computers internal clock.

DateToJulian

Returns the Julian date of input value. The input value must be a valid date (January 1, 1900 to February 28, 2100). The value returned will be a number from 1 to 73049.

Category:

Date & Time

Function:

DateToJulian(num)

Example:

DateToJulian(860617) returns 31578

DayOfMonth

The DayOfMonth function returns a number (3-31) representing the day of the month of the study day.

Category:

Date & Time

Function:

DayOfMonth(num)

Example:

DayOfMonth(901023) returns a value of 23

DayOfWeek

The DayOfWeek function returns a day number representing the day of the week of the trading day identified by studyday. The return value will be from 0-6, as follows:

0 = Sunday

1 = Monday

2 = Tuesday

3 = Wednesday

4 = Thursday

5 = Friday

6 = Saturday

Category:

Date & Time

Function:

DayOfWeek(num)

Example:

DayofWeek(900719) returns 4

JulianToDate

Returns the date in YYMMDD format of the input Julian date. This input value may range from 1 (January 1, 1900) through 73049 (February 28, 2100).

Category:

Date & Time

Function:

JulianToDate(num)

Example:

JulianToDate(31578) returns 860617

Month

The Month function returns the number of the month of studyday. The number returned is between 3-12.

1 = January 7 = July

2 = February 8 = August

3 = March 9 = September

4 = April 10 = October

5 = May 11 = November

6 = June 12 = December

Category:
Date & Time
Function:
Month(num)
Example:
Month(921120) returns 11

Product

Returns a number representing the Omega Research product you are working with.
10 = TradeStation, TradeStation Real Time and TradeStation Special Edition
23 = SuperCharts Real Time
11 = SuperCharts
Category:
Date & Time
Function:
Product(num)

Year

The Year function returns the last two digits of the year number.
Category:
Date & Time
Function:
Year(num)
Example:
Year(901120) returns 90

Drawing Objects Category

The functions in the **Drawing Objects category** are functions that allow you to draw text objects and trendlines to the screen and then manipulate them. They also enable you to access and manipulate existing text objects and trendlines.

These functions can only be accessed by means of the PowerEditor; they are not available when using the QuickEditor.

Note: When using trendline functions (they begin with TL_), keep in mind that some return a value and some do not. You can have all of them return values if you want the PowerEditor to return an error code if and when the function is not successful. However, having the function return a value when not necessary will slow down the performance of your function.

The function for which you want a detailed explanation:

GetBackgroundColor

Category:
Drawing Object
Function:
GetBackgroundColor
Usage:
Returns the color to which the chart windows background is currently set, 1 - 16. Color values are listed in the section Color Values. If the function was not successful, a negative number error code is returned. Error codes are listed in the section Error Codes.
Example:
If the current chart windows background color is set to dark blue, using GetBackgroundColor would return 9, which is equal to Tool_DarkBlue.

Text_Delete

Category:
Drawing Object
Function:
Text_Delete(TX_Ref)
Usage:
Deletes a text object from the chart window. If the function is not successful, an error code is returned.
Example:
Text_Delete(92) deletes text object number 92 from the chart.

Text_GetDate

Category:

Drawing Object

Function:

Text_GetColor(TX_Ref)

Usage:

Returns the color of the specified text object. The numeric values and colors to which they are assigned are listed in the section Color Values. If the function is not successful, returns an error code.

Example:

Value1 = Text_GetColor(56) returns the color of the text object number 56.

Text_GetColor

Category:

Drawing Object

Function:

Text_GetDate(TX_Ref)

Usage:

Returns the date axis value of the specified text object. If the function is not successful, returns an error code.

Example:

Value1 = Text_GetDate(56) returns the date axis value set in the text object number 56.

Text_GetFirst

Category:

Drawing Object

Function:

Text_GetFirst(OwnedPref)

Usage:

Returns the object reference number for the oldest (first created) text object of the requested ownership type on the chart. The ownership types are as follows:

1 = first created by current analysis technique

2 = first created by user or other technique

3 = first created by all techniques

Note: Input values other than 1, 2 or 3 will cause the function to assume an input value of 3. If the function was unsuccessful, or there are no text objects on the chart, an error code is returned.

Example:

Say 3 is a specific indicator, using Value1 = Text_GetFirst(3) returns the reference for the indicators first text object.

Text_GetHStyle

Category:

Drawing Object

Function:

Text_GetHStyle(TX_Ref)

Usage:

Returns the horizontal text placement style for a text object. Horizontal styles are listed in the section Text Style Codes. If the function is not successful, an error code is returned.

Example:

Value1 = Text_GetHStyle(51) returns the horizontal style for text object number 51.

Text_GetNext

Category:

Drawing Object

Function:

Text_GetNext(TX_Ref,OwnedPref)

Usage:

Returns the object reference number for the text object of the requested ownership type created immediately after the text object passed as a reference variable. The ownership types are as follows:

- 1 = first created by current analysis technique
- 2 = first created by user or other thechnique
- 3 = first created by all techniques

Note: Input values other than 1, 2 or 3 will cause the function to assume an input value of 3. If the function was unsuccessful, or there are no text objects created on the chart after the one passed in as a variable, an error code is returned.

Example:

Value1 = Text_GetNext(18,1) returns the reference for the first text object created after text object number 18.

Text_GetString

Category:

Drawing Object

Function:

Text_GetString(TX_Ref)

Usage:

Returns the text of the specified text object. This can be returned into a string-type variable or passed into any string-type function input. If the function is not successful, it will return an empty string of length = 0.

Example:

Text_GetString(56) returns the text stored in text object number 56.

Text_GetTime

Category:

Drawing Object

Function:

Text_GetTime(TX_Ref)

Usage:

Returns the time axis value of the specified text object. If the function is not successful, returns an error code.

Example:

Value1 = Text_GetTime(56) returns the time axis value of the text object number 56.

Text_GetValue

Category:

Drawing Object

Function:

Text_GetValue(TX_Ref)

Usage:

Returns the price axis value set in the specified text object. If the function is not successful, returns an error code.

Example:

Value1 = Text_GetValue(56) returns the price axis value set for text object number 56.

Text_GetVStyle

Category:

Drawing Object

Function:

Text_GetVStyle(TX_Ref)

Usage:

Returns the vertical text placement style for the specified text object. Vertical styles are listed in the section Text Style Codes. If the function is not successful, an error code is returned.

Example:

Value1 = Text_GetVStyle(18) returns the vertical placement style for text object number 18.

Text_New

Category:

Drawing Object

Function:

Text_New(Date,Time,Val,str)

Usage:

Creates and draws a new text object on the active chart window, in the location specified. You must specify the date, time, value and string to add. The date and time inputs make up what can be considered the x coordinates and the value input can be considered the y coordinate. The charting application will find the coordinates closest to the ones specified. The text you pass can be any type of easy language string.

The return value is the text box reference number (referred to as TX_Ref), otherwise it will return an error code.

Example:

Value1 = Text_New(950225,1100,23, UP) places the text object Up on the chart and returns the text box reference number.

Text_SetColor

Category:

Drawing Object

Function:

Text_SetColor(TX_Ref,Color)

Usage:

Changes the color of the specified text object from its original color to the specified color. If the function is not successful, an error code is returned.

Example:

Text_SetColor(51,Tool_Red) turns the color of text object number 51 from its original color to red.

Text_SetLocation

Category:

Drawing Object

Function:

Text_SetLocation(TX_Ref,Date,Time,Val)

Usage:

Moves a text object to a new location in the chart window specified by means of the date, time and value specified. The date and time inputs can be considered the x-coordinates and the value can be considered the y-coordinate. If the function is not successful, an error code is returned.

Example:

Text_SetLocation(24,950418,1100,32) moves text object number 24 to the new location. The new location will be where April, 18, 1995 at 11am intersects with the price 32.

Text_SetString

Category:

Drawing Object

Function:

Text_SetString(TX_Ref,Str)

Usage:

Changes the text of the specified text object. If the function is not successful, an error code is returned.

Example:

Text_SetString(43, MovAvg) changes the text in text object number 43 to MovAvg.

Text_SetStyle

Category:

Drawing Object

Function:

Text_SetStyle(TX_Ref,hStyle,vStyle)

Usage:

Enables you to set the text placement style code for a text object. The vertical and horizontal style codes are listed in the section Text Style Codes. If the function is not successful, an error code is returned.

Example:

Text_SetStyle(24,2,0) sets text object 24 centered and below the price line.

TL_Delete

Category:

Drawing Object

Function:

TL_Delete(TL_Ref)

Usage:

Deletes the specified TrendLine and its reference.

Example:

TL_Delete(19) deletes TrendLine number 19 from the active chart window.

TL_GetAlert

Category:

Drawing Object

Function:

TL_GetAlert(TL_Ref)

Usage:

Returns a number between 0 and 2, indicating the type of alert associated with the specified trendline. Alert types are listed in the section Alert Values. If the function is not successful, the function returns an error code.

Example:

Value1 = TL_GetAlert(34) returns the alert type for trendline number 34.

TL_GetBeginDate

Category:

Drawing Object

Function:

TL_GetBeginDate(TL_Ref)

Usage:

This function obtains the beginning date of the given trendline and returns it in YYMMDD format. For example, if the trendline begins on August 23, 1995, the function will return 950823. If the function is not successful, the function will return an error code.

Example:

Value1 = TL_GetBeginDate(7) returns the date at which trendline number 7 begins.

TL_GetBeginTime

Category:

Drawing Object

Function:

TL_GetBeginTime(TL_Ref)

Usage:

This function obtains the beginning time of the given trendline and returns it in military time format. For example, if the trendline begins at 11:24am, the function will return 1124, and if it begins at 3:35pm, it will return 1535. If the function is not successful, the function will return an error code.

Example:

Value1 = TL_GetBeginTime(7) returns the time of the bar at which trendline number 7 begins.

TL_GetBeginVal

Category:

Drawing Object

Function:

TL_GetBeginVal(TL_Ref)

Usage:

Returns the value of the price axis (y-coordinate) at the TrendLines start point. If the function is not successful, the function will return an error code.

Example:

Value1 = TL_GetBeginVal(7) returns the price axis at the start point of TrendLine number 7.

TL_GetColor

Category:

Drawing Object

Function:

TL_GetColor(TL_Ref)

Usage:

Returns the specified TrendLines color number. For a list of the colors and their numeric values, please refer to the section Color Values. If the function is not successful, the function will return an error code.

Example:

Value1 = TL_GetColor(7) returns the color number of TrendLine number 7. For example, if the color of TrendLine number 7 is magenta, the function would return a 5, which is Tool_Magenta.

TL_GetEndDate

Category:

Drawing Object

Function:

TL_GetEndDate(TL_Ref)

Usage:

This function obtains the ending date of the given trendline and returns it in YYMMDD format. For example, if the trendline ends on August 23, 1995, the function will return 950823. If the function is not successful, the function will return an error code.

Example:

Value1 = TL_GetDate(7) returns the date of the end point for TrendLine number 7.

TL_GetEndTime

Category:

Drawing Object

Function:

TL_GetEndTime(TL_Ref)

Usage:

This function obtains the ending time of the given trendline and returns it in military time format. For example, if the trendline ends at 11:24am, the function will return 1124, and if it ends at 3:35pm, it will return 1535. If the function is not successful, the function will return an error code.

Example:

Value1 = TL_GetEndTime(7) returns the bar time at the end-point of TrendLine number 7.

TL_GetEndVal

Category:

Drawing Object

Function:

TL_GetEndVal(TL_Ref)

Usage:

Returns the price value at the specified TrendLines end point. If the function is not successful, the function will return an error code.

Example:

Value1 = TL_GetEndVal(7) returns the value at the end point of TrendLine number 7.

TL_GetExtLeft

Category:

Drawing Object

Function:

TL_GetExtLeft(TL_Ref)

Usage:

Returns TRUE if the specified trendline has been extended to the left. If the trendline has not been extended, returns FALSE.

Example:

If trendline number 18 has been extended to the left, using Value1 = TL_GetExtLeft(18) would return the expression TRUE.

TL_GetExtRight

Category:

Drawing Object

Function:

TL_GetExtRight(TL_Ref)

Usage:

Returns TRUE if the specified trendline has been extended to the right. If the trendline has not been extended, returns FALSE.

Example:

If TrendLine number 18 has been extended to the right, using Value1 = TL_GetExtRight(18) would return the expression TRUE.

TL_GetFirst

Category:

Drawing Object

Function:

TL_GetFirst(OwnedPref)

Usage:

Returns the object reference number for the oldest (first created) trendline of the requested ownership type on the chart. The ownership types are as follows:

- 1 = first created by current analysis technique
- 2 = first created by user or other thechnique
- 3 = first created by all techniques

Note: Input values other than 1, 2 or 3 will cause the function to assume an input value of 3.

If the function was unsuccessful, or there are no trendlines on the chart, an error code is returned.

Example:

Value1 = TL_GetFirst(3) returns the reference for the first TrendLine in the chart window with the specified ownership type.

TL_GetNext

Category:

Drawing Object

Function:

TL_GetNext(TL_Ref,OwnedPref)

Usage:

Returns the object reference number for the trendline of the requested ownership type created immediately after the trendline passed as a reference variable. The ownership types are as follows:

- 1 = first created by current analysis technique
- 2 = first created by user or other thechnique
- 3 = first created by all techniques

Note: Input values other than 1, 2 or 3 will cause the function to assume an input value of 3.

If the function was unsuccessful, or there are no trendlines created on the chart after the one passed in as a variable, an error code is returned.

Example:

Value1 = TL_GetNext(18,1) returns the trendline created immediately after trendline number 18.

TL_GetSize

Category:

Drawing Object

Function:

TL_GetSize(TL_Ref)

Usage:

Returns a value, 0-6, represending the thickness of the referenced TrendLine. Thickness values are listed in the section Line Thickness Values. If the function is not successful, an error code is returned.

Example:

Value1 = TL_GetSize(18) returns the thickness of TrendLine number 18. If TrendLine 18 is set to the thinnest line style possible, it returns a 0.

TL_GetStyle

Category:

Drawing Object

Function:

TL_GetStyle(TL_Ref)

Usage:

Returns the number equivalent to the specified TrendLines style. For a list of the styles and their numeric values, please refer to the section Style Values. If the function is not successful, the function will return an error code.

Example:

If the line style of TrendLine number 7 is dashed, then using Value1 = TL_GetStyle(7) would return a 2, which is Tool_Dashed.

TL_GetValue

Category:

Drawing Object

Function:

TL_GetValue(TL_Ref,Date,Time)

Usage:

This function finds the y-axis value that the referenced trendline will cross for the given date and time. If successful, returns a positive value (price/y-coordinate); if unsuccessful, returns an error code. Error codes are listed in the section Error Codes.

Example:

Value1 = TL_GetValue(14,950123,1400) returns the price at 2pm on January 23, 1995 based on TrendLine number 14.

TL_New

Category:

Drawing Object

Function:

TL_New(sDate,sTime,sVal,eDate,eTime,eVal)

Usage:

Creates a new TrendLine with the specified starting date, starting time, starting value, ending date, ending time and ending value. The date and time inputs make up what can be considered the x coordinates, and the value inputs can be considered the y coordinates. The charting application will find the closes matching time to the one specified, and if sDate is greater than eDate or sTime greater than eTime (on same date), will switch the date and/or time inputs to reflect the proper time sequence.

The return value is the trendline reference number if successful; otherwise, an error code is returned. Error codes are listed in the section Error Codes.

Example:

Value1 = TL_New(Date1,1600,Price1,Date2,1600,Price2) creates a TrendLine starting at Date1, 1600, Price1 and ending at Date2, 1600, Price2.

TL_New can be specified on any data stream: Value1 = TL_New(sD,sT,sV,eD,eT,eV) Data3; or on the default data stream Data1.

TL_SetAlert

Category:

Drawing Object

Function:

TL_SetAlert(TL_Ref,AlertState)

Usage:

Changes a trendlines alert setting to one of 3 types. The available alerts are listed in the section Alert Values.

Example:

TL_SetAlert(7,2) sets the alert of trendline number 7 to trigger on bar close (type 2).

TL_SetBegin

Category:

Drawing Object

Function:

TL_SetBegin(TL_Ref,sDate,sTime,sVal)

Usage:

Changes the start point of the specified TrendLine.

Note: When a trendline is moved such that the end date falls after the previous begin date, you must set the begin date (using TL_SetBegin function) before setting the new end date.

Example:

TL_SetBegin(8,950221,1100,Close) changes the start point of TrendLine number 8 to the point at February 21, 1995, 11am, on the Close price.

TL_SetColor

Category:

Drawing Object

Function:

TL_SetColor(TL_Ref,Color)

Usage:

Changes the color of the specified TrendLine. Either a color name or a color value can be passed as the color input. The available colors are listed in the section Color Values.

Example:

TL_SetColor(53,Tool_Black) changes the color of TrendLine number 53 to black.

TL_SetEnd

Category:

Drawing Object

Function:

TL_SetEnd(TL_Ref,eDate,eTime,eVal)

Usage:

Changes the end point of the specified TrendLine.

Note: When a trendline is moved such that the start date falls after the previous end date, you must set the end date (using TL_SetEnd function) before setting the new begin date.

Example:

TL_SetEnd(4,951207,1300,Close) changes the end point of TrendLine number 4 to the point at December 7, 1995 at 1pm on the Close price.

TL_SetExtLeft

Category:

Drawing Object

Function:

TL_SetExtLeft(TL_Ref,tfExtend)

Usage:

Specifies whether to extend a TrendLine to the left.

Example:

TL_SetExtLeft(5,FALSE) disables the left extend on TrendLine number 5.

TL_SetExtRight

Category:

Drawing Object

Function:

TL_SetExtRight(TL_Ref,tfExtend)

Usage:

Specifies whether to extend a TrendLine to the right.

Example:

TL_SetExtLeft(5,FALSE) disables the right extend on TrendLine number 5.

TL_SetSize

Category:

Drawing Object

Function:

TL_SetSize(TL_Ref,Size)

Usage:

Changes the thickness of a trendline to a specified width. Thickness values 0-6 are passed as the thickness input. Thickness values are listed in the section Line Thickness Values.

Example:

TL_SetSize(10,0) sets trendline number 10 to the thinnest line width, which is 0.

TL_SetStyle**Category:**

Drawing Object

Function:

TL_SetStyle(TL_Ref,Style)

Usage:

Changes the line style of the specified TrendLine. A style name or value can be passed as the style input. Style names and values are listed in the section Style Values.

Example:

TL_SetStyle(5,Tool_Dashed3) changes the line style of TrendLine number 5 to the dashed-3 style.

Style Values

The following color values are returned by certain functions in the Drawing Objects category:

Tool_Solid 1
Tool_Dashed 2
Tool_Dotted 3
Tool_Dashed24
Tool_Dashed35

Error Codes

The following error codes are returned by the functions in the Drawing Object category when the function was not successful:

- 2 The object identifier was invalid
- 3 The data number (Data2, Data3, etc.) passed to the function was invalid (probably won't be able to get since PowerEditor will catch an invalid reference before study can be applied).
- 4 The value passed to a SET function was invalid.
- 5 The beginning and ending points were the same.
- 6 The function was unable to load the default values for the tool.
- 7 Unable to add the object. Possibly due to an out of memory condition.
- 8 Invalid pointer.
- 9 Previous failure.
- 10 Too many trendline objects on the chart.
- 11 Too many text objects on the chart.

Alert Values

The following alert values are returned by certain functions in the Drawing Objects category:

No alert 0
Intra-Bar alert 1
Alert on close 2

Color Values

The following color values are returned by certain functions in the Drawing Objects category:

Tool_Black 1
Tool_Blue 2
Tool_Cyan 3
Tool_Green 4
Tool_Magenta 5
Tool_Red 6
Tool_Yellow 7
Tool_White 8
Tool_DarkBlue 9
Tool_DarkCyan 10
Tool_DarkGreen 11
Tool_DarkMagenta 12
Tool_DarkRed 13
Tool_DarkYellow 14
Tool_DarkGray 15
Tool_LightGray 16

Line Thickness Values

The following line thickness values are returned by certain functions in the Drawing Objects category:		
Thinnest		0
Very Thin	1	
Thin	2	
Medium		3
Thick	4	
Very Thick	5	
Thickest		6
<u>Text Style Codes</u>		
The following text style codes are returned by certain functions in the Drawing Objects category:		
<u>Date-line = horizontal/x-coordinate</u>		
Hright	0	Drawn right of the date-line
Hleft	1	Drawn left of the date-line
Hcenter	2	Drawn centered on the date-line
<u>Price-line = vertical/y-coordinate</u>		
Vbelow	0	Drawn below the price-line
Vabove	1	Drawn above the price-line
Vcenter	2	Drawn centered on the price-line

Easy Language Category

Easy Language functions are the most commonly-used functions and serve the purpose of indicating the direction and strength of the market. They are written in Omega Researchs Easy Language and are often used to write even more intricate rules. All the functions in the category Easy Language are described next, in alphabetical order.

The listing for each function includes a detailed description of the functions purpose and use as well as a description of the value returned by the function.

The function for which you want a detailed explanation:

AccumDist

The Accumulation Distribution function examines total daily volume in an effort to identify the beginning of a Bull or Bear market move. Signals are interpreted by examining the divergence between the accumulation distribution value for the current bar and the underlying asset price.

For example, a buy signal could be interpreted when the underlying assets price reaches a new High without a similar new High in the accumulation distribution value.

This is a cumulative function, meaning the current bars value is influenced by what has happened in the past. Also, total volume is used as opposed to trade volume. Total volume is reported once a day, at the end of the day.

Category:

Easy Language

Function:

AccumDist(MYVOL)

Parameters:

MYVOL Total volume of current bar, at the end of the day

Returns:

Accumulation distribution value for the current bar

Usage:

MYVOL is the only parameter of the function. Traditionally, this function uses total volume or some form of volume as the MYVOL parameter. In most cases, when you use this function, you'll use the Easy Language Volume field to replace MYVOL, which is the total daily volume for the symbol. However, you can use an input to replace MYVOL.

It is important to realize that for most futures contracts, the days volume is not reported until the following day. In these cases, you will always be one day behind. Once the data is updated at the end of the day, the previous days volume is corrected. For stocks and most stock indexes, volume is reported without this time lag. Notice that the words today and yesterday and not this bar or last bar are being used. This is because this study uses total volume which is only available once a day.

Example:

You could write an indicator that has three plot lines: the first plots the Accumulation Distribution with a lag time of one day, the second plot line uses a two-day lag time, and the last plot line uses a three-day lag time. Lag time refers to plotting yesterdays values today for the first plot, the value of two days ago for the second plot, and the value of three days ago for the third plot. You would use the Offset parameter do accomplish this, as shown in

the figure below.

Accumulation Distribution function used with hard-coded Offset parameter

With this indicator, you would be trying to answer two questions. The first question is whether the Accumulation Distribution technical study has historically been able to provide an indication as to the direction and/or strength of the market. The second question is whether the study has a lag time associated with it. That is, do we get a better indication of the markets direction by looking at the value of the study from two or three days ago rather than looking at more recent values?

You could have also made the number of offset days a variable, thereby testing different offset days. For example, the variable Lag1 could have replaced the input 1, the variable Lag2 could have replaced the input 2, and the variable Lag3 could have replaced the input 3. The two figures below illustrate this PaintBar study.

In order to use the offset variables, the setting for the Maximum number of bars referenced by a study (referred to as MaxBarsBack) must be set to a value equal to or greater than the greatest offset value. For example, if your largest offset value is 7, the smallest your MaxBarsBack value could be is 7.

AccumSwingIndex

The Accumulation Swing Index function is directly related to the Swing Index function. The Accumulation Swing Index keeps a running total of the SwingIndex values for a current bar. The SwingIndex function was developed to help cut through the maze of Open, High, Low and Close prices to indicate the real strength and direction of the market. The Swing Index function looks at the Open, High, Low and Close values for a two-bar period. The theory is that there are four cross-bar and one intra-bar comparisons that are strong indicators of an up or down day.

The Swing Index returns a number between -100 and 100. If the factors point toward an up day, then the function value will be positive and vice versa. In this way, the Swing Index gives us definite short-term swing points, and it can be used to supplement other methods as a breakout indicator. A breakout is indicated when the value of the Accumulation Swing Index (ASI) exceeds the ASI value on the day when a previous significant High Swing Point was made. A downside breakout is indicated when the value of the ASI drops below the ASI value on a day when a previous significant low swing point was made.

Since only futures have a relative daily limit value, this function only makes sense when applied to a futures contract. If you use this function and it only plots a zero flat line, check the Daily Limit value.

Category:

Easy Language

Function:

AccumSwingIndex

Parameters:

None

Returns:

Accumulation Swing Index value for the current bar, either a negative or positive value.

Usage:

If the long term trend is up, the ASI will be a positive value. If the long-term trend is down, the ASI will be a minus value. For additional information, please refer to the Swing Index function.

Example:

Please refer to the example given for the Swing Index function.

ADX

The ADX function attempts to measure the trending quality of the market, isolating those periods where the market is not trending.

One of the problems with trend-following systems is that you must endure the agony of a choppy market while you are waiting for a trend. If the ADX can truly tell us when the market is trending, we should be able to eliminate trades during this choppy period by filtering our trades with the ADX. The ADX tells you whether the market is moving directionally or not. It does not pick direction, rather it simply indicates whether or not, and by how much the market is trending.

The greater the ADX value, the more a market is trending.

Category:

Easy Language

Function:

ADX(LENGTH)

Parameters:

LENGTH the number of trailing bars to consider in the ADX function

Returns:

ADX value for the current bar

Usage:

The ADX function is a moving average of technical analyst Welles Wilders study, DX. In Omega Researchs Easy Language, the function DMI is the equivalent of DX. DMI is based on two component functions: DMI Plus and DMI Minus.

The ADX function has a single parameter, LENGTH. The parameter LENGTH refers to the number of bars used to calculate the moving average of DX. The parameter should be replaced by a positive whole number or a simple numeric input.

ADXCustom

The ADXCustom, like the ADX function, attempts to measure the trending quality of the market, isolating those periods where the market is not trending. In fact, the ADXCustom function is identical to the ADX function except that the ADXCustom function provides greater flexibility, you can specify what value to use when calculating the DMI functions.

The ADX function calls the DMI Plus and DMI Minus functions, whereas the ADXCustom function calls the DMI Plus Custom and DMI Minus Custom functions. The DMI Plus and DMI Minus functions calculate values called DM Plus and DM Minus. The standard DMI Plus and DMI Minus functions use the High, Low, and Close prices of the bar to calculate DM Plus and DM Minus. The ADXCustom function gives you the added flexibility of being able to specify what value the DM Plus and DM Minus values are calculated on.

Category:

Easy Language

Function:

ADXCustom(HPRICE,LPRICE,CPRICE,LENGTH)

Parameters:

HPRICE High price

LPRICE Low price

CPRICE Close price

LENGTH specifies the number of trailing bars to consider

Returns:

ADXCustom value for the current bar

Usage:

Most custom functions are used like the standard versions of the function with the exception of being able to alter the values on which the function calculates its value. For example, with the ADXCustom function you can replace the parameters HPRICE, LPRICE, and CPRICE with a valid Easy Language expressions, you are not limited to the actual High, Low, and Close prices. Many clients like to assign values using multiple data series, as illustrated in

the example below. Also, remember that HPRICE, LPRICE, and CPRICE can all be replaced by numeric simple or numeric series type inputs.

ADX

The ADXR function relates all symbols by rating them according to their movement. The ADXR function returns a value between zero and one hundred. The higher the number the greater the movement. As with the value returned by the ADX function, this number will not identify the direction of the trend, only the strength of the movement.

Since the ADX function is based on the DMI, it will have excessive variance at tops and bottoms. To compensate for the variance, you can use a 14-day differential of the ADX as the final rating factor for directional movement. This final rating factor is called the average direction movement index rating (ADXR). The ADXR function takes the ADX value of the current bar, adds the ADX value of (LENGTH - 1) bars ago to it, then divides that sum by two.

The theory is that while volatility can be an indicator of movement, movement does not necessarily indicate volatility. For this reason, the most important index to use for a trend-following system is the ADXR; however, most money is generally made in the shortest period of time when the stock or commodity is volatile. Those who don't like the risk of volatile markets should use the ADXR function.

Category:

Easy Language

Function:

ADXR(MYRANGE)

Parameters:

MYRANGE the number of trailing bars to consider

Returns:

ADXR value for the current bar

Usage:

The parameter MYRANGE refers to the length of the moving average used in the ADX function, which is called by the ADXR function. The parameter can be replaced with a constant positive whole number or by a simple numeric input. Overlaying two plot lines, the ADX and the ADXR, could reveal interesting relationships between the two.

ADXRCustom

Most custom functions are identical to the standard versions of the function, with the one exception of being able to specify the values on which the function calculates. The standard ADXR function calls the ADX function, which in turn calls the DMI Plus and DMI Minus functions. The ADXRCustom function calls the ADXCustom function, which in turn calls the DMI Plus Custom and DMI Minus Custom functions.

Again, the ADXRCustom function gives you the flexibility of specifying what value the DM Plus and DM Minus values are to be calculated upon.

Category:

Easy Language

Function:

ADXRCustom(HPRICE,LPRICE,CPRICE,LENGTH)

Parameters:

MYRANGE the number of trailing bars to consider

Returns:

ADXRCustom value for the current bar

Usage:

Traditionally, the DMI Plus and DMI Minus functions calculate values called DM Plus and DM Minus. The standard DMI Plus and DMI Minus functions use the High, Low, and Close prices of the bar to calculate DM Plus and DM Minus respectively.

Remember, the HPRICE, LPRICE, and CPRICE values can all be written directly into the code or replaced by numeric simple or numeric series type inputs.

Average

The Average function is a simple average of the prices for the last x bars. Moving averages are probably the most well known among technical indicators. The Average function is often used to smooth the values of other functions or values.

Category:

Easy Language

Function:

Average(PRICE,LENGTH)

Parameters:

PRICE the data series to average

LENGTH the trailing number of bars to average

Returns:

Mean average value for the current bar

Usage:

The parameter PRICE can be hard-coded with a value such as Close or (High + Low)/2, or can be replaced with a variable for greater flexibility. The value plugged in for the PRICE parameter does not necessarily need to be a price. For example, you may want to average the value of the RSI function for the last 24 bars. To do this you would use the Average function and then plug in the RSI function in place of the PRICE parameter

The parameter LENGTH tells the function how many bars to average. The value plugged into this parameter should be a positive whole number, or a numeric input.

AvgPrice

This function returns the average price of a bar. The average price is calculated by adding The Open, High, Low, and Close, and then dividing the sum by four. If the Open is not available in the data, then the calculation consists of summing the High, Low, and Close and then dividing by three.

Category:

Easy Language

Function:

AvgPrice

Parameters:

None

Returns:

Value of $(O+H+L+C)/4$, or $(H+L+C)/3$ if Open not available

Usage:

Using the result of this function instead of any single price element, such as the Close, has the effect of more accurately representing the movement of the entire bar. It also smoothes the effect of extreme Highs and Lows in the price bar. A popular use of the Average Price function is to plug it into the PRICE parameter of other functions such as the RSI and Rate of Change.

AvgTrueRange

In his book, *New Concepts in Technical Trading Systems*, Welles Wilder, Jr. uses a value known as true range in his directional movement calculations. He defines true range as the larger of the following:

The distance between today's High and today's Low.

The distance between today's High and yesterday's Close, or

The distance between today's Low and yesterday's Close.

AverageTrueRange is simply an average of the True Range values over a period of time.

Category:

Easy Language

Function:

AvgTrueRange(LENGTH)

Parameters:

LENGTH number of bars in the average

Returns:

AverageTrueRange value, the average of the True Range of LENGTH bars

Usage:

The AverageTrueRange tends to peak during periods of higher than average volatility and to bottom out during periods of lower than average volatility. The market tends to top out when Plot1 hits a new 18-bar High. The market tends to bottom out when the AverageTrueRange hits a new 18-bar Low.

BarNumber

The BarNumber function is very useful in designing studies and systems. Each bar on a chart (after the number of bars specified by the Maximum number of bars referenced by a study (known as MaxBarsBack) is assigned a number, which is incremented by 1 with each

successive bar. For example, if your MaxBarsBack is presently set to 10, the 11th bar has a BarNumber of 1, the 12th bar has a BarNumber of 2, and so on.

Category:

Easy Language

Function:

BarNumber

Parameters:

None

Returns:

Current bar number

Usage:

The BarNumber function can be easily confused with another function called CurrentBar. The main difference between these two functions is that CurrentBar references only the most recent bar, whereas, BarNumber can reference any bar.

Many times an event will occur on a particular bar and you'll want to be able to identify other characteristics of that bar at a later point.

BearishDivergence

The BearishDivergence function searches for and finds occurrences of highs in prices (i.e., High, Close, etc.) not accompanied by highs in an oscillator (i.e., RSI, Percent R, SlowD). If the criteria of the Bearish Divergence function is met, a +1 will be returned; if it has not been met, a 0 will be returned.

The Bearish Divergence function calls the function SwingHighBar. The SwingHighBar function returns the bar number of the most recent occurrence of the pattern shown in the figure below. It can also be set to return the second, third, fourth, etc., most recent occurrence of a Swing High.

Category:

Easy Language

Function:

BearishDivergence(PRICE,OSC,STRENGTH,LENGTH)

Parameters:

PRICE specifies which price of the asset to use

OSC indicator applied to the asset of interest

STRENGTH required number of bars on either side of swing bar

LENGTH the number of trailing bars to consider

Returns:

True/False - If the condition(s) are found true, the function returns a positive one (1). If the condition(s) are found false the function returns a zero (0).

Usage:

The four parameters of the BearishDivergence function are tied to two SwingHighBar functions called by the BearishDivergence function. The parameters PRICE, STRENGTH, and LENGTH are used by one of the SwingHighBar function, and the parameters OSC, STRENGTH, and LENGTH are used in the other.

The PRICE and OSC parameters tell the SwingHighBar function what to use in its calculations. PRICE is usually replaced by a price element of a bar (i.e., Close, High, Low) while OSC is usually replaced by an oscillator (i.e., RSI, Percent R, SlowD).

STRENGTH refers to the number of bars on either side of the Swing High bar. The higher the STRENGTH parameter, the longer the Swing High pattern takes to be identified. For example, a Swing High pattern with a strength of 1 needs three bars to identify the pattern and will not be known until the bar after it occurred. A Swing High pattern with a strength of 3 needs seven bars to identify the pattern and will not be known until three bars after it occurred.

LENGTH tells the function how many bars to analyze at a time. If no swing highs are found in that number of bars, the function returns a -1 and moves on to the next bar. For example, if you had 50 bars of data and your LENGTH was 10, bars one through ten would be tested initially, then bars two through eleven, then bars three through twelve . . . then bars forty-one through fifty.

Note: Usually, increasing the LENGTH and STRENGTH of the BearishDivergence function will decrease the number of occurrences.

The bars with the black dot in the figure below are Swing High bars based on the High with a STRENGTH of 1 and a LENGTH of 5. Notice the High on either side of the Swing High bar

is less than the High of the Swing High bar.

Swing High

By using the SwingHighBar function, the BearishDivergence function finds the bar numbers of the most recent and second most recent occurrence of a Swing High, and returns the values of the PRICE and OSC parameters for those bars.

It then compares the most recent and second most recent occurrences based on the PRICE parameter, checking to see if the most recent occurrence is greater than the second most recent occurrence. It also compares the most recent and second most recent occurrences based on OSC. It checks to see if the value of the most recent occurrence is less than the value of the second most recent occurrence.

If both of these conditions are found true the function returns a +1. If either or both conditions are found false the function returns a 0.

BlackScholes

The Black Scholes option pricing model calculates an approximate fair value for a European style option given the user specified inputs.

Category:

Easy Language

Function:

BlackScholes(DAYSLEFT,STRIKE,LEVEL,RATE1,VOLA,TYPE)

Parameters:

DAYSLEFT days remaining until the relevant option expiration date

STRIKE strike price of option contract

LEVEL current price of the underlying security

RATE current short-term interest rate, entered as whole number

VOLA volatility of the underlying option

TYPE Put = 0; Call = 1

Usage:

The DaysLeft input can be calculated using the DaysToExpiration function. Also, it is very difficult to determine the value to plug into the VOLA input. Therefore, it is highly recommended that the VOLA input value be derived from the value returned by the Implied Volatility function. Once the value is obtained, plug it into the VOLA input. Then the value returned by the Black Scholes function can be used to determine if the price of the option is oversold or overbought.

BollingerBand

Trading bands are among the most widely used technical indicators. They are lines drawn at fixed intervals (usually a percentage) around a moving average. Bollinger Bands are bands

that vary in distance from the moving average based on volatility. The upper band represents x number of standard deviations above the average, and the lower band represents x number of standard deviations below the average. By using standard deviations rather than a fixed percentage, the bands adjust for volatility. During volatile periods, the bands move further away from the average, while during market lulls, the bands move closer to the average.

The closer the prices move to the upper band, the more overbought the market, and the closer the prices move to the lower band, the more oversold the market.

Category:

Easy Language

Function:

BollingerBand(PRICE,LENGTH,SDEV)

Parameters:

PRICE specifies which price of the asset to use

LENGTH number of trailing bars to average

SDEV number of standard deviations to offset

Returns:

Bollinger Band value for the current bar

Usage:

In practice, the band offsets (standard deviations) should be adjusted to enclose about 90 percent of market activity, with about 10 percent slipping either above or below the Bollinger Bands. This allows for an accurate assessment of the true trading range of the underlying security.

In addition, some traders may want to experiment using a larger offset for the lower band, to allow for the tendency of down trends to be sharper and of shorter duration than up trends. While the conventional usage always has the security rated as either overbought or oversold, a better method is to allow for a neutral reading when prices fall about half way between the upper and lower bands. Applying this method, a security would only be rated as overbought or oversold if it were actually very close to or beyond either the upper or lower band.

In addition, significant consideration should be given to the price momentum. A security which has entered into overbought or oversold territory may continue to become even more overbought or oversold before reversing. For this reason, look for evidence of price weakness/strength before expecting a reversal.

BullishDivergence

The BullishDivergence function finds occurrences of when there are lows in prices (i.e., Low, Close, etc.) not accompanied by lows in an oscillator (i.e., RSI, Percent R, SlowD).

Category:

Easy Language

Function:

BullishDivergence(PRICE,OSC,STRENGTH,LENGTH)

Parameters:

PRICE specifies which price of the asset to use

OSC the indicator applied to the asset of interest

STRENGTH required number of bars on either side of swing bar

LENGTH the number of trailing bars to consider

Returns:

True/False

Usage:

If the criteria of the function is met, a +1 is returned; if the criteria is not met, a 0 is returned.

The Bullish Divergence function calls another function SwingLowBar. The SwingLowBar function returns the bar number of the most recent occurrence of the pattern as shown in Figure 6-15. It can also be set to return the second, third, fourth, etc. most recent occurrence of a Swing Low.

All parameters for the BullishDivergence function are tied to two SwingLowBar functions called by the BullishDivergence function. The parameters PRICE, STRENGTH, and LENGTH are used by one of the SwingLowBar function, while the parameters OSC, STRENGTH, and LENGTH are used in the other.

The PRICE and OSC parameters tell the SwingLowBar function on what to perform its calculations. PRICE is usually replaced by a price element of a bar (i.e., Close, High, Low), while OSC is usually replaced by an oscillator (i.e., RSI, Percent R, SlowD).

STRENGTH refers to the number of bars on either side of the Swing Low bar. The higher the STRENGTH, the longer the Swing Low pattern takes to be identified. For example, a Swing Low pattern with a strength of 1 needs 3 bars to identify the pattern and will not be known until the bar after it occurred. A Swing Low pattern with a strength of 3 needs 7 bars to identify the pattern and will not be known until three bars after it occurred.

LENGTH tells the function how many bars to analyze at a time. If no Swing Lows are found in that number of bars the function returns a -1 and moves on to the next bar. For example, if you had 50 bars of data and your length was 10, bars one through ten would be tested initially, then bars two through eleven, then bars three through twelve . . . then bars forty-one through fifty.

Note: Usually, increasing the LENGTH and the STRENGTH of the BullishDivergence function will decrease the number of occurrences.

The bars with the black dot in the figure below are Swing Low bars based on the Low with a STRENGTH of 1 and a LENGTH of 5. Notice the Low on either side of the Swing Low bar is less than the Low of the Swing Low bar.

Swing Low

By using the SwingLowBar function, the BullishDivergence function finds the bar numbers of the most recent and second most recent occurrences of a Swing Low and returns the values of the PRICE and OSC parameters for those bars.

It then compares the most recent and second most recent occurrences based on the PRICE parameter, checking to see if the most recent occurrence is less than the second most recent occurrence. It also compares the most recent and second most recent occurrences based on OSC. This time it checks to see if the value of the most recent occurrence is greater than the value of the second most recent occurrence.

If both of these conditions are found true, the function returns a +1. If either or both conditions are found false, the function returns a 0.

CalcTime

The CalcTime function adds and subtracts minutes from the time entered; it takes the time entered and adds or subtracts the number of minutes entered and returns the new time, in HHMM format.

Category:

Easy Language

Function:

CalcTime(XTIME, MINUTES)

Parameters:

XTIME time to which minutes will be added or subtracted, entered in HHMM format

MINUTES minutes to be added (positive value) or subtracted (negative value)

Usage:

The time entered in XTIME must be entered in HHMM format. To add minutes, enter a positive value in MINUTES, and to subtract minutes, enter a negative value in MINUTES.

Call

The Call function can be used as a substitute or input for the parameter TYPE in many of the Option indicators (i.e., Black Scholes, OptionProfile, etc.). The TYPE parameter must be replaced by a 0 or a 1, to indicate either a Put or a Call. Using this function or the Put function eliminates the need to remember the correct number to enter. Instead, you can just enter the word Call or Put.

Category:

Easy Language

Function:

Call

Parameters:

None

Usage:

This function always returns a 1, to indicate a Call.

CCI

The Commodity Channel Index (CCI), which is best used with contracts which display cyclical or seasonal characteristics, is formulated to detect the beginning and ending of these cycles by incorporating a moving average together with a divisor which reflects both possible and actual trading ranges. The final index measures the deviation from normal which indicates major changes in market trend.

Category:

Easy Language

Function:

CCI(LENGTH)

Parameters:

LENGTH the number of trailing bars to consider

Returns:

CCI value for the current bar

Usage:

The LENGTH parameter tells the function how many bars to analyze at a time. For example, if you had 50 bars of data, and your length was ten, bars one through ten would be tested first, then bars two through eleven, and so on. A CCI value will be calculated once for each bar.

The CCI value usually does not fall outside the -300 to 300 range and, in fact, is usually in the -100 to 100 range. Traditionally, when the CCI function is used in a system, a long position is taken when CCI exceeds 100 and a short position when CCI falls below -100.

Note: Many traders use this indicator the opposite way. They use it as an overbought/oversold indicator with 100 or greater indicating that the market is overbought, and -100 or less that the market is oversold.

ChaikinOsc

This indicator looks to improve on Larry William's Accumulation Distribution. Williams' formula compared the closing price with the opening price. In the early 1970's, opening prices for stocks stopped being transmitted by the exchanges. This made it difficult to calculate Williams' formula. The Chaikin Oscillator uses the average price of the bar calculated as follows $(High + Low) / 2$ instead of the Open.

The indicator subtracts a 10 period exponential moving average of the AccumDist function from a 3 period exponential moving average of the AccumDist function. The indicator also identifies the current trend as follows:

$Average(Close, 9) > Average(Close, 9)[1]$ and $Close \geq Average(Close, 9)$ then the current trend is up.

$Average(Close, 9) < Average(Close, 9)[1]$ and $Close < Average(Close, 9)$ then the current trend is down.

Note: If neither of the preceding two conditions are met, the trend is the same as it was on the previous bar.

Category:

Easy Language

Function:

ChaikinOsc(MYVOL, FAST, SLOW)

Parameters:

MYVOL volume (the volume of the day)

FAST period used to calculate the fast exponential moving average

SLOW period used to calculate the slow exponential moving average

Usage:

When price reaches a new High and the oscillator fails to do the same, this is seen as a bearish signal. When price reaches a new Low and the oscillator fails to do the same, this is seen as a bullish signal. Those signals that occur in the same direction as the current trend are more reliable than those that do not.

When the oscillator is above zero and has just reversed downward after increasing for several bars, this is seen as a bearish signal. When the oscillator is below zero and has just reversed upward after decreasing for several bars, this is seen as a bullish signal. Those signals that occur in the opposite direction of the current trend are more reliable than those that do not.

Correlation

Correlation is defined as a measure of the tendency of two variables to be above or below their means at the same time. The Correlation function returns a value between +1 and -1. A value close to +1 indicates a strong positive relationship while a value close to -1 indicates a strong inverse relationship. A value of 0 indicates no correlation.

Category:

Easy Language

Function:

Correlation(IND,DEP,LENGTH)

Parameters:

IND specifies the independent variable

DEP specifies the dependent variable

LENGTH the number of trailing bars to consider

Returns:

Value in the range -1 to +1. A value close to +1 = strong positive relationship; a value close to -1 = strong inverse relationship; a value of 0 = no correlation.

Usage:

The parameters IND and DEP stand for the independent and dependent variable, respectively, and can be replaced with any valid Easy Language expression, a numeric simple input or a numeric series input.

This function is most beneficial when comparing two data streams with one another. For example, if the price of oats goes up, there is little doubt that the price of feeder cattle will as well. In this example, oats is the independent variable and feeder cattle is the dependent variable. In other words, the price of feeder cattle is affected by the price of oats.

CSI

The CSI function attempts to identify those markets that are more likely to provide greater returns for each dollar invested. The factors used in the calculation are directional movement, volatility, margin requirement, and commissions.

Category:

Easy Language

Function:

CSI(MRGN,COMM,LENGTH)

Parameters:

MRGN specifies the margin requirement

COMM specifies the commission amount

LENGTH the number of trailing bars to consider

Returns:

CSI value for the current bar

Usage:

The factors are individually weighted, with directional movement having the highest weight, volatility having the second highest, margin requirement having the third highest, and commission having the least weight. The higher the CSI value the higher the rating the security receives.

The formula used to calculate the CSI value for the current bar is as follows:

Refer to the ADXR function earlier in this chapter for a better understanding on how the ADXR function is calculated. The ATR is calculated by summing the TR (True Range) for n bars and then dividing by n. The TR for a bar is derived from the largest value of the following:

The distance between todays High and todays Low

The distance between todays High and yesterdays Close

The distance between todays Low and yesterdays Close

By using the CSI function, you will only have to plug in the commission, margin, and length (n) the rest is taken care of for you.

Cum

This function is designed to add the parameter INPUT1 beginning from the bar after the number of bars specified by the Maximum number of bars referenced by a study setting (known as MaxBarsBack) all the way up to the current bar.

Category:

Easy Language

Function:

Cum(INPUT1)

Parameters:

INPUT1 specifies the data series to accumulate

Returns:

Cumulative value up to and including the current bar

Usage:

For example, if you replace the parameter INPUT1 with Close, the value on the last bar of data would represent the sum of all Close prices starting from the bar after the bar specified by the Maximum number of bars referenced by a study setting (known as MaxBarsBack) up to the current bar. You can replace the parameter with either a numeric simple or numeric series function.

DailyLosers

The DailyLosers function returns the number of losing positions that were taken throughout the date specified by the input TRGTDATE.

Category:

Easy Language

Function:

DailyLosers(TRGTDATE)

Parameters:

TRGTDATE date you want to check for losing trades (YYMMDD format)

Usage:

This function uses the function PositionProfit, which can only refer back to the profit of the last 11 trades (including the current open position). Therefore, if more than 11 trades were taken during a day, this function will only account for the number of losing trades that occurred in the last 11.

Based on your trading tendencies, the value returned by this function can persuade you to cease all trading for the day or continue trading.

DailyWinners

The DailyWinners function returns the number of winning trades that were taken throughout the date specified by the input TRGTDATE.

Category:

Easy Language

Function:

DailyWinners(TRGTDATE)

Parameters:

TRGTDATE date you want to check for winning trades (YYMMDD format)

Usage:

This function uses the function PositionProfit, which can only refer back to the profit of the last 11 trades (including the current open position). Therefore, if more than 11 trades were taken during a day, this function will only account for the number of winning trades that occurred in the last 11.

Based on your trading tendencies, the value returned by this function can persuade you to cease all trading for the day or continue trading.

DarkCloud

The DarkCloud function determines an average length of the body and evaluates the current and the previous bars. The function compares the body of the previous bar to the average body, and then looks at the position of the current bar to determine whether or not the current bar completes a DarkCloud candlestick formation.

Category:

Easy Language

Function:

DarkCloud(Length)

Parameters:

Length the number of trailing bars used in determining the average length of the body.

Returns:

True/False

DayOfWeekFix

This function returns the day of the week based on the date entered.

Category:

Easy Language

Function:

DayOfWeekFix(TDATE)

Parameters:

TDATE date for which to determine day of week

Usage:

This function is used when you want to determine the day of the week on which a particular date fell or will fall.

DaysToExpiration

The DaysToExpiration function returns the number of days left between today's date and the specified stock options expiration date.

Category:

Easy Language

Function:

DaysToExpiration(EXMONTH, EXYEAR)

Parameters:

EXMONTH expiration month of option

EXYEAR expiration year of option

Usage:

This function can be used with the BlackScholes function, in place of the DaysLeft parameter. The number of days left to expiration is an important factor to consider when pricing options.

Delta

The Delta function returns the expected change in the option price for a one-point change in the underlying security.

Category:

Easy Language

Function:

Delta(DAYSLEFT, STRIKE, LEVEL, RATE, VOLA, TYPE)

Parameters:

DAYSLEFT number of days left to the expiration date of option

STRIKE exercise price of option

LEVEL price of the underlying asset

RATE currently available risk-free rate of return (enter 5% as 5)

VOLA annual volatility of underlying asset

TYPE Put = 0; Call = 1

Usage:

The Delta function measures the expected change in an option price resulting from a +1 point change in the price of the underlying asset. The delta of a call is positive, while the delta of a put is negative.

Example:

Say a MRK May 40 Call trading at 1 has a delta of .375, while MRK is trading at 37. If MRK were to rise in price by 1 point, to 38, the put would be expected to rise in price by .375 from 1 to 1.375.

Delta call values range from 0 to 1. The more the option is in-the-money, the closer to 1 delta becomes.

Delta is also commonly referred to as the hedge ratio, as it is used to calculate a hedge. For example, to hedge 10,000 shares of IBM sold short, consider the following option: IBM June 80 Put with a delta of .25

Delta allows you to calculate the number of option contracts required to hedge this position. The short IBM shares have a delta of $-1 * 10,000$. That is, for every dollar IBM rises, the short position loses \$10,000 in value.

Therefore, to hedge this stock position (to create a position that neither gains nor loses money due to price fluctuations in IBM) we need to determine the number of put contracts necessary to gain \$10,000 from a 1 point rise in IBM.

Since 1 option controls 100 shares, a delta of .25 means that the options contract will increase in value by \$25 for a 1 point rise. Therefore, 400 contracts ($\$10,000 / \25) are needed to create this hedge.

DMI

The DMI function is based on values obtained by the DMIPlus and DMIMinus functions. The value of the DMI function is a number between 0 and +100. The higher the DMI, the more directional the movement. Higher values are associated with more directional movement, and indicate a stronger trending quality. The lower the DMI value, the less directional the movement. The DMI does not indicate the direction of the trend, only the quality of the movement.

Category:

Easy Language

Function:

DMI(LENGTH)

Parameters:

LENGTH the number of trailing bars to consider

Returns:

DMI value for the current bar

Usage:

The function takes the absolute value of the difference between DMI Plus and DMI Minus, divides that value by the sum of DMI Plus and DMI Minus, then multiplies that quotient by

100. This formula is shown below:

The parameter LENGTH is used in both the DMI Plus and DMI Minus functions, which are both called by the DMI function. The parameter can be replaced with a number or by a numeric simple variable.

DMICustom

The DMI Custom function is based on values obtained by the DMI Plus Custom and DMI Minus Custom functions. The value of the DMI Custom function is a number between 0 and +100. The higher the DMI Custom value, the more directional the movement. Higher values are associated with more directional movement, and indicate a stronger trending quality. The lower the DMI Custom value, the less directional the movement. The DMI Custom function does not indicate the direction of the trend, only the quality of the movement.

Most custom functions are used like the standard versions of the function, except that with the custom function, you have the ability to alter the values on which the function calculates. For example, with the DMICustom function, you can replace the parameters HPRICE, LPRICE, and CPRICE with a valid Easy Language expressions or with numeric series inputs. Many clients like to assign values using multiple data series.

Category:

Easy Language

Function:

DMICustom(HPRICE,LPRICE,CPRICE,LENGTH)

Parameters:

LENGTH the number of trailing bars to consider

Returns:

DMI custom value for the current bar

Usage:

The function takes the absolute value of the difference between DMIPlusCustom and DMIMinusCustom, divides that value by the sum of DMIPlusCustom and DMIMinusCustom,

then multiplies that quotient by 100. The formula is as follows:

The HPRICE, LPRICE, and CPRICE are additional parameters not found in the standard DMI function. The input LENGTH, is used in the same way it is used in the standard DMI function.

DMIMinus

Most traders use DMI Minus in the following way: when DMI Minus crosses above the DMI Plus, establish a short position. The DMI Minus value is always positive. Some traders create an oscillator, which is DMI Plus - DMI Minus. A positive value represents an up-trend and a negative value represents a downtrend.

Category:

Easy Language

Function:

DMIMinus(LENGTH)

Parameters:

LENGTH the number of trailing bars to consider

Returns:

DMI Minus value for the current bar

Usage:

To obtain the DMIMinus, all DM- for x (LENGTH) bars would be added together and divided by the sum of the true ranges for x (LENGTH) bars. The true range of a bar is determined by choosing the largest of the following:

The distance between today's High and today's Low

The distance between today's High and yesterday's Close

The distance between today's Low and yesterday's Close

The DMIMinus function is not typically used by itself for any significant market analysis; rather it is more often used with the DMIPlus function. When the DMIPlus line is above the DMIMinus line, the market is usually considered bullish, and when DMIPlus is below DMIMinus, it is considered bearish.

The DMIMinus function has one parameter, LENGTH, which can be hard coded with a number or an expression that results in a number or it can be replaced with an numeric simple type input.

DMIMinusCustom

Most custom functions are used like the standard versions of the function except that with the custom function you can alter the values on which the function calculates. For example, with the standard DMIMinus function calculations are made using the High, Low, and Close of the bar. With the DMIMinusCustom function, you are not limited to the High, Low, and Close prices; you can replace the parameters HPRICE, LPRICE, and CPRICE with valid Easy Language expressions or numeric series type inputs. Many clients like to assign values using multiple data series.

Category:

Easy Language

Function:

DMIMinusCustom(HPRICE,LPRICE,CPRICE,LENGTH)

Parameters:

HPRICE High price

LPRICE Low price

CPRICE Close price

LENGTH the number of trailing bars to consider

Returns:

DMI Minus custom value for the current bar

Usage:

HPRICE, LPRICE, and CPRICE are additional parameters not found in the standard DMIMinus function. The parameter LENGTH is used in the same way it is used in the standard DMIMinus function.

DMIMinusCustom is the Easy Language equivalent of DI- in technical analyst Welles Wilders formula. DI- is the quotient of something known as DM- divided by True Range of a defined number of bars. DM stands for directional movement and can be negative (DM-) or

positive (DM+). The directional movement is determined by looking at the current and the previous bars, as shown below:

In pattern #1 the current bars HPRICE and LPRICE are higher than the previous bars HPRICE and LPRICE. Therefore, the directional movement is positive.

In pattern #2 the current bars HPRICE and LPRICE is lower than the previous bars HPRICE and LPRICE. Therefore, the directional movement is negative.

In pattern #3, the HPRICE and LPRICE of the current and previous bar are identical, therefore there is no directional movement.

Pattern #4 is an outside bar pattern. With outside bars there can be either a positive or negative movement depending on which value is greater. In our example DM+ was greater than DM-.

Pattern #5 is an inside bar pattern. All inside bar patterns have no directional movement.

To obtain the DMIMinusCustom, all DM- for x (LENGTH) bars would be added together and divided by the sum of the true ranges for x (LENGTH) bars. The true range of a bar is determined by choosing the largest of the following:

The distance between today's HPRICE and today's LPRICE

The distance between today's HPRICE and yesterday's CPRICE

The distance between today's LPRICE and yesterday's CPRICE

The DMIMinusCustom function is not typically used by itself for any significant market analysis; rather it is more often used with the DMIPlusCustom function. When the DMIPlusCustom line is above the DMIMinusCustom line, the market is usually considered bullish, and when DMIPlusCustom is below DMIMinusCustom, it is considered bearish.

DMIPlus

Most traders use the DMIPlus in this way: when DMIPlus crosses above DMIMinus, establish a long position. Other traders create an oscillator which is DMI Plus - DMI Minus. A positive value represents an up-trend, and a negative value represents a downtrend.

Category:

Easy Language

Function:

DMIPlus(LENGTH)

Parameters:

LENGTH the number of trailing bars to consider

Returns:

DMI Plus value for the current bar

Usage:

DMIPlus is the Easy Language equivalent of DI+ in technical analyst Welles Wilders formula. DI+ is the quotient of something known as DM+ divided by the True Range for a defined number of bars. DM stands for directional movement and can be negative (DM-) or positive (DM+). The directional movement is determined by looking at the current and the previous bars. The patterns are shown below:

In pattern #1 the current bars High and Low are higher than the previous bars High and Low. Therefore, the directional movement is positive.

In pattern #2 the current bars High and Low is lower than the previous bars High and Low. Therefore, the directional movement is negative.

In pattern #3, the High and Low of the current and previous bar are identical, therefore there is no directional movement.

Pattern #4 is an outside bar pattern. With outside bars there can be either a positive or negative movement depending on which value is greater. In our example DM+ was greater than DM-.

Pattern #5 is an inside bar pattern. All inside bar patterns have no directional movement.

To obtain the DMIPlus, all DM+ values for x (LENGTH) bars would be added together and divided by the sum of the true ranges for x (LENGTH) bars. The true range of a bar is determined by choosing the largest of the following:

The distance between today's High and today's Low

The distance between today's High and yesterday's Close

The distance between today's Low and yesterday's Close

The DMIPlus function is not typically used by itself for any significant market analysis; rather it is more often used with the DMIMinus function. When the DMIPlus line is above the DMIMinus line the market is usually considered bullish and when DMIPlus is below DMIMinus it is considered bearish. The DMIPlus function has one input LENGTH which can be hard coded with a number or an Easy Language expression that returns a number or it can be replaced with a numeric simple type input.

DMIPlusCustom

Most custom functions are much like the standard versions of the function except that the custom functions allow you the flexibility of being able to alter the values on which the function calculates. For example, with the standard DMIPlus function, calculations are made using the High, Low, and Close prices of the bar. With the DMIPlusCustom function, you can replace the parameters HPRICE, LPRICE, and CPRICE with any valid Easy Language expression or numeric series type inputs. Many clients like to assign values using multiple data series.

Category:

Easy Language

Function:

DMIPlusCustom(HPRICE,LPRICE,CPRICE,LENGTH)

Parameters:

HPRICE High price

LPRICE Low price

CPRICE Close price

LENGTH the number of trailing bars to consider

Returns:

DMI Plus Custom value for the current bar

Usage:

The parameters HPRICE, LPRICE, and CPRICE are additional parameters not found in the standard DMIPlus function. The parameter LENGTH is used in the same way it is used in the standard DMIPlus function.

DMIPlusCustom is the Easy Language equivalent of DI+ in technical analyst Welles Wilders formula. DI+ is the quotient of something known as DM+ divided by True Range of a defined number of bars.

DM stands for directional movement and can be negative (DM-) or positive (DM+). The directional movement is determined by looking at the current and the previous bars.

In pattern #1 the current bars HPRICE and LPRICE are higher than the previous bars HPRICE and LPRICE. Therefore, the directional movement is positive.

In pattern #2 the current bars HPRICE and LPRICE is lower than the previous bars HPRICE and LPRICE. Therefore, the directional movement is negative.

In pattern #3, the HPRICE and LPRICE of the current and previous bar are identical, therefore there is no directional movement.

Pattern #4 is an outside bar pattern. With outside bars there can be either a positive or negative movement depending on which value is greater. In our example DM+ was greater than DM-.

Pattern #5 is an inside bar pattern. All inside bar patterns have no directional movement.

To obtain the DMIPlusCustom, all DM+ for x (LENGTH) bars would be added together and divided by the sum of the true ranges for x (LENGTH) bars. The true range of a bar is determined by choosing the largest of the following:

The distance between today's HPRICE and today's LPRICE

The distance between today's HPRICE and yesterday's CPRICE

The distance between today's LPRICE and yesterday's CPRICE

The DMIPlusCustom function is not typically used by itself for any significant market analysis; rather it is more often used with the DMIMinusCustom function. When the DMIPlusCustom line is above the DMIMinusCustom line the market is usually considered bullish and when DMIPlusCustom is below DMIMinusCustom it is considered bearish.

Doji

The Doji function evaluates the current bar and determines whether or not the difference between the bars Open and Close prices are within an acceptable range.

Category:

Easy Language

Function:

Doji(Tolerance)

Parameters:

Tolerance the acceptable range between the Open and Close prices of a bar, represented as a percentage of the bars total range.

Returns:

True/False

EaseOfMovement

This indicator gauges the magnitude of price and volume movement. The indicator returns both positive and negative values. A positive value means the market has moved up from yesterday's value, whereas, a negative value means the market has moved down. A large positive or large negative value indicates a large move in price and/or lighter volume. A small positive or small negative value indicates a small move in price and/or heavier volume.

Category:

Easy Language

Function:

EaseOfMovement(LENGTH)

Parameters:

LENGTH Time period of the EaseOfMovement function

Returns:

The indicator returns both positive and negative values. A positive value means the market has moved up from yesterday's value, whereas, a negative value means the market has moved down.

Usage:

When the Ease of Movement indicator crosses above zero, the market is moving upward easily; buy opportunity. When the indicator crosses below zero, it is an indication that the market is moving downward easily; usually a sell opportunity.

EntriesToday

Returns the number of entries that have been taken on the date specified by parameter TRGTDATE.

Category:

Easy Language

Function:

EntriesToday(TRGTDATE)

Parameters:

TRGTDATE date to check for total entries, entered in YYMMDD format

Usage:

This function uses the function EntryDate, which can only store the information for the last ten entries (excluding the current open position). Therefore, this function will return a maximum value of 11. Based on your trading tendencies, the value returned by this function can persuade you to cease all trading for the day or continue trading.

EveningStar

The EveningStar function determines an average length of the body and evaluates the current and the two previous bars. The function compares the body of the bar preceeding the previous bar to the average body, and then looks at the position of the previous and the current bar to determine whether or not the current bar completes a EveningStar candlestick formation.

Category:

Easy Language

Function:

EveningStar(Length)

Parameters:

Length the number of trailing bars used in determining the average length of the body.

Returns:

True/False

ExitsToday

Returns the number of exits that have been taken on the date specified by parameter TRGTDATE.

Category:

Easy Language

Function:

ExitsToday(TRGTDATE)

Parameters:

TRGTDATE date to check for total exits, entered in YYMMDD format

Usage:

This function uses the function ExitDate, which can only store the information for the last ten exits. Therefore, if more than 10 exits occurred, the maximum number this function will return is 10. Based on your trading tendencies, the value returned by this function can persuade you to cease all trading for the day or continue trading.

FastD

This is a stochastic function. Stochastics are oscillators that are used to indicate overbought and oversold conditions in the market based on the premise that during periods of price decreases, bar closes tend to accumulate near the Low of the bar, and during periods of price increases, bar closes tend to accumulate near the High of the bar.

Category:

Easy Language

Function:

FastD(LENGTH)

Parameters:

LENGTH indicates number of bars used in the calculation

Usage:

Please refer to the discussion under the Stochastics function.

FastDCustom

This is a stochastic function. Stochastics are oscillators that are used to indicate overbought and oversold conditions in the market based on the premise that during periods of price decreases, bar closes tend to accumulate near the Low of the bar, and during periods of price increases, bar closes tend to accumulate near the High of the bar.

Category:

Easy Language

Function:

FastDCustom(HPRICE, LPRICE, CPRICE, LENGTH)

Parameters:

HPRICE High price to be used in calculation

LPRICE Low price to be used in calculation

CPRICE Closing price to be used in calculation

LENGTH indicates number of bars used in the calculation

Usage:

Please refer to the discussion under the Stochastics function.

FastHighestBar

This function is used internally by the HighestBar function and should not be called directly by the user. The user should always use HighestBar instead.

Category:

Easy Language

Function:

FastHighestBar(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH the number of trailing bars to consider

Returns:

The number of bars ago that the highest value found occurred

FastK

This is a stochastic function. Stochastics are oscillators that are used to indicate overbought and oversold conditions in the market based on the premise that during periods of price decreases, bar closes tend to accumulate near the Low of the bar, and during periods of price increases, bar closes tend to accumulate near the High of the bar.

Category:

Easy Language

Function:

FastK(LENGTH)

Parameters:

LENGTH indicates number of bars used in the calculation

Usage:

Please refer to the discussion under the Stochastics function.

FastKCustom

This is a stochastic function. Stochastics are oscillators that are used to indicate overbought and oversold conditions in the market based on the premise that during periods of price decreases, bar closes tend to accumulate near the Low of the bar, and during periods of price increases, bar closes tend to accumulate near the High of the bar.

Category:

Easy Language

Function:

FastKCustom(HPRICE, LPRICE, CPRICE, LENGTH)

Parameters:

HPRICE High price to be used in calculation

LPRICE Low price to be used in calculation

CPRICE Closing price to be used in calculation

LENGTH indicates number of bars used in the calculation

Usage:

Please refer to the discussion under the Stochastics function.

FastLowestBar

This function is used internally by the LowestBar function and should no be called directly by the user. The user should always use either LowestBar as an alternative.

Category:

Easy Language

Function:

FastLowestBar(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH the number of trailing bars to consider

Returns:

The number of bars ago the lowest value found occurred

FindBar

The Find Bar function searches back in time for the first bar matching the date and time specified through the parameters TRGTDATE and TRGTTIME. It searches back through the full range of bars specified in the Maximum number of bars referenced by a study setting (known as MaxBarsBack).

Category:

Easy Language

Function:

FindBar(TRGTDATE, TRGTTIME)

Parameters:

TRGTDATE date of the bar to find, entered in YYMMDD format

TRGTTIME time of the bar to find, entered in 24-hour military format

Usage:

This function uses the MRO function, but forces the MRO function to keep the parameters as numeric simple arguments. On its own, the MRO function forces parameters to be numeric series arguments, which unnecessarily increases the size of your study or system.

Gamma

The Gamma function measures the rate of change of Delta for a one-point change in the price of the underlying security.

Category:

Easy Language

Function:

Gamma(DAYSLEFT, STRIKE, LEVEL, RATE, VOLA, TYPE)

Parameters:

DAYSLEFT number of days left to the expiration date of option

STRIKE exercise price of option

LEVEL price of the underlying asset

RATE currently available risk-free rate of return (enter 5% as 5)

VOLA annual volatility of underlying asset

TYPE Put = 0; Call = 1

Hammer

The Highest function is designed to return the highest value found when applying the parameter PRICE over a period of time defined by the parameter LENGTH.

Category:

Easy Language

Function:

Highest(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH the number of trailing bars to consider

Returns:

The highest value found

Usage:

To understand the function we must understand its parameters. The parameter PRICE is usually hard coded with some bar attribute such as Close, Open, High, Low, and Volume or is replaced with a numeric series type input. However, it can also be replaced with a valid Easy Language expression. For example, Close + Open or Average(RSI(Close,14),14).

The parameter LENGTH, just like PRICE, can be hard coded or can be replaced with a numeric simple type input. The value is usually a positive whole number such as 5, 10, 14 etc. Once again you may choose to replace LENGTH with a valid numeric expression. If you do decide to make it a numeric expression, keep in mind that LENGTH should be a positive whole number and cannot change on a bar-to-bar basis as can the value PRICE.

For the sake of simplicity, let's look at how the function works with Close for the PRICE and 14 for LENGTH. For each bar, the function returns the highest Close over the previous 14

trailing bars. There may be times when more two or more bars tie for the highest value. When this happens, the function returns the PRICE for the most recent bar.

A common usage of the Highest function is to find a breakout. For example, an important occurrence is when the value of the PRICE parameter for the current bar is the highest value over the last x bars. Many people encounter a problem when using the function to find a breakout because they forget to remove the current bar from the function.

The statement they usually enter is `Close>Highest(Close, 14)`. The problem with this statement is that it will never be evaluated as true. Lets assume that the Close of the current bar was also the highest Close over the last fourteen bars, this would only equate the two sides of the expression. Therefore to overcome this problem you must exclude the last bar from the calculation of the function. This is done by offsetting the function by one bar. Thus, you will be comparing the Close of the current bar with the value returned by the Highest function for the previous bar. The corrected statement is `Close>Highest(Close,14)[1]`.

HangingMan

The HangingMan function evaluates the current bar as well as the market trend (bullish or bearish) and determines whether or not the candlestick formation is a HangingMan formation.

Category:

Easy Language

Function:

`HangingMan(Length,Tail)`

Parameters:

Length the number of trailing bars to consider in determining up- or down-trend.

Tail the length of the body (the distance from the open to the close) is multiplied by this number to determine the minimum acceptable length of the tail (the distance from the low to the body).

Returns:

True/False

Highest

The Highest function is designed to return the highest value found when applying the parameter PRICE over a period of time defined by the parameter LENGTH.

Category:

Easy Language

Function:

`Highest(PRICE,LENGTH)`

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH the number of trailing bars to consider

Returns:

The highest value found

Usage:

To understand the function we must understand its parameters. The parameter PRICE is usually hard coded with some bar attribute such as Close, Open, High, Low, and Volume or is replaced with a numeric series type input. However, it can also be replaced with a valid Easy Language expression. For example, `Close + Open` or `Average(RSI(Close,14),14)`.

The parameter LENGTH, just like PRICE, can be hard coded or can be replaced with a numeric simple type input. The value is usually a positive whole number such as 5, 10, 14 etc. Once again you may choose to replace LENGTH with a valid numeric expression. If you do decide to make it a numeric expression, keep in mind that LENGTH should be a positive whole number and cannot change on a bar-to-bar basis as can the value PRICE.

For the sake of simplicity, lets look at how the function works with Close for the PRICE and 14 for LENGTH. For each bar, the function returns the highest Close over the previous 14 trailing bars. There may be times when more two or more bars tie for the highest value. When this happens, the function returns the PRICE for the most recent bar.

A common usage of the Highest function is to find a breakout. For example, an important occurrence is when the value of the PRICE parameter for the current bar is the highest value over the last x bars. Many people encounter a problem when using the function to find a breakout because they forget to remove the current bar from the function.

The statement they usually enter is `Close>Highest(Close, 14)`. The problem with this statement is that it will never be evaluated as true. Lets assume that the Close of the current bar was also the highest Close over the last fourteen bars, this would only equate the two

sides of the expression. Therefore to overcome this problem you must exclude the last bar from the calculation of the function. This is done by offsetting the function by one bar. Thus, you will be comparing the Close of the current bar with the value returned by the Highest function for the previous bar. The corrected statement is `Close>Highest(Close,14)[1]`.

HighestBar

The HighestBar function is very similar to the Highest function. Whereas, the Highest function returns the value of the highest PRICE parameter, the HighestBar function returns the number of bars ago it occurred.

Category:

Easy Language

Function:

`HighestBar(PRICE,LENGTH)`

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH the number of trailing bars to consider

Returns:

The number of bars ago the highest value found occurred

Usage:

To understand the function, we must understand its parameters. The parameter PRICE is usually hard coded with some bar attribute such as Close, Open, High, Low, and Volume or is replaced with a numeric series type input. However, it can be replaced with a valid Easy Language expression. For example: `Close + Open`, or `Average(RSI(Close,14),14)`.

The parameter LENGTH, just like PRICE, can be hard coded replaced with an numeric simple type input. The value is usually a whole positive number such as 5, 10, 14, etc. Once again, you may choose to replace LENGTH with a valid Easy Language expression. If you do decide to make it an expression, keep in mind that LENGTH should be a positive whole number and cannot change on a bar-to-bar basis as can PRICE.

For the sake of simplicity, let us look at how the function works with the hard coded values Close for PRICE and 14 for LENGTH. On each bar, the function returns the number of bars ago on which the highest Close was achieved over the last 14 bars.

There may be times when more two or more bars tie for the highest value. When this happens the function identifies the most recent bar.

HPI

The Herrick Payoff Index function is used to analyze futures and commodities. The value returned by the function is a measure of the money flow in and out of the market or commodity to which it is applied.

Category:

Easy Language

Function:

`HPI(MULT,FACTOR)`

Parameters:

MULT the contract value of a 1 cent move in the underlying asset

FACTOR a user-defined smoothing constant

Returns:

HPI value for the current bar

Usage:

Open Interest is a requirement in the Herrick Payoff Index, therefore, this study is only

applicable to daily data for futures, options, and commodities data. The formula applied is as follows:

...where K_y = Yesterdays HPI

S = User-entered smoothing factor

C = the value of a 1 cent move

V = Volume

I = The absolute value of today's open interest or yesterday's open interest, whichever is greater.

G = the greater of today's or yesterday's open interest.
M = High minus Low and the difference is divided by two.

My = Yesterday's High minus yesterday's Low and the difference is divided by two.
The formula has a minus sign below a plus sign. If $M > My$ the plus sign is used, however, if $M < My$ the minus sign is used.

When the Herrick Payoff Index function is called by any analysis technique, it will require two parameters from the user. The first parameter, MULT is the value of a 1 cent move, the letter C in the equation. This value will depend on the security that is loaded. For example, if the security being loaded was cattle this value would be \$400. If the security being loaded was soybeans this value would be \$50.

The second parameter, FACTOR, is the user entered smoothing factor, the letter S in the equation. The FACTOR more or less corresponds to a moving-average time span. Higher values used for the smoothing factor tend to give more reliable results. Noted technician Tom Aspray, in a March 1988 article in Stocks and Commodities magazine, recommends a value of 21.

To interpret the Herrick Payoff Index, determine whether the Index is above or below zero. Readings above zero indicate that interest in the market is growing. This is regarded as a positive sign (bullish). On the other hand, readings below zero can be interpreted as a negative sign (bearish).

Another important occurrence in the Index is when it moves to an extreme, then reverses quite sharply. This indicates that the price reached a short-term buying or selling climax.

The Index often shows whether a rally signaled by other short term indicators will turn out to be worthwhile or not.

IFF

The IFF function is used for conditional statements, and is one of the most important functions because indicators and functions are designed to calculate values, and in the case of indicators also plot those values they are not designed to perform conditional checks. However, by using the IFF function, you are able to check one or more conditions and if the COND parameter is true, take one action, and if the COND parameter is false, take another action.

Category:

Easy Language

Function:

IFF(COND,TVALUE,FVALUE)

Parameters:

COND expression to check (i.e., Close > Open)

TVALUE value if COND expression is true

FVALUE value if COND expression is false

Returns:

TVALUE if COND is true; FVALUE if COND is false

Usage:

This function has three parameters. The first parameter is COND which stands for condition. This refers to a true/false conditional statement.

An example of a conditional statement is: Is Today's bar an outside bar? If it is an outside bar, then the statement is true. If it is not an outside bar, then the statement is false.

The second parameter is TVALUE, which refers to the action you want to take if the parameter COND is evaluated as being true. The third parameter is FVALUE, which refers to the action you want to take if the parameter COND is evaluated as being false.

ImpliedVolatility

The Implied Volatility function returns the market implied volatility for the specified Option.

Category:

Easy Language

Function:

ImpliedVolatility(EXMONTH, EXYEAR, STRIKE, RATE, OPPRICE, TYPE, ASSET)

Parameters:

EXMONTH month of Options expiration

EXYEAR year of Options expiration

STRIKE exercise price of the Option

RATE currently available risk-free interest rate

OPPRICE price of the option

TYPE	Put = 0; Call = 1
ASSET	price of the underlying asset

LastCalcDate

The LastCalcDate function returns the date for the last completed bar, in YYMMDD format

Category:

Easy Language

Function:

LastCalcDate

Parameters:

None

Usage:

For example, if the function returns 960203, it means the last bar was completed on 02/03/96, or February 3, 1996.

LastCalcTime

The LastCalcTime function returns the time of completion (close) of the last bar, in 24-hour military format.

Category:

Easy Language

Function:

LastCalcTime

Parameters:

None

Usage:

For example, if the function returns 1700, the last bar was completed, or closed, at 5:00pm.

Leader

The Leader function is a simple function that compares the current and previous bars to determine if the mid-point of the current bar is greater than the previous High or less than the previous Low.

If either condition is true, the function returns a 1; if neither is true, then it returns a 0.

Category:

Easy Language

Function:

Leader

Parameters:

None

Usage:

The code or formula for this function is as follows:

Value1 = (High + Low) / 2 ;

Cond1 = Value1 > High[1] OR Value1 < Low[1] ;

If Cond1 then

Leader = 1

else

Leader = 0 ;

LinearRegAngle

Linear Regression is a concept also known as the least squares method or best fit. Linear Regression attempts to fit a straight line between several data points in such a way that distance between each data point and the line is minimized.

This function is very similar to the LinearRegSlope function, except it returns the angle of that line in terms of degrees. In fact, to obtain the LinearRegAngle one could take the ArcTangent of the LinearRegSlope function.

Category:

Easy Language

Function:

LinearRegAngle(PRICE,LEN)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LEN specifies the number of trailing bars to consider

Returns:

The angle of the current regression line

Usage:

When the function is used in a study or system, it requires two parameters from the user, PRICE and LEN. The PRICE parameter tells the function which data points to use.

For example, if you want the Linear Regression Angle based on the Close, replace PRICE with Close. The PRICE parameter of the function can also be replaced with a variable. For example, if you wanted to calculate the linear regression of the Relative Strength Index, you could replace the PRICE parameter with RSI(Close,14).

The LEN parameter tells the function how many bars to consider in the regression. The parameter must be hard coded with a number like 5 or replaced with a numeric simple input. Numeric simple inputs limit you to simple expressions that return a number that doesn't change on a bar-to-bar basis. At this time, all numeric simple type parameters can not use any of the functions.

In addition, the value returned for the parameter LEN should be a positive whole number since it refers to the number of bars.

LinearRegSlope

Linear Regression is a concept also known as the least squares method or best fit. Linear Regression attempts to fit a straight line between several data points in such a way that distance between each data point and the line is minimized.

This function is very similar to the LinearRegAngle function, except it returns the slope of that regression line instead of the angle.

Category:

Easy Language

Function:

LinearRegSlope(PRICE,LEN)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LEN the number of trailing bars to consider

Returns:

The slope of the current regression line

Usage:

When the function is used in a study or system it requires two parameters from the user, PRICE and LEN. The PRICE parameter tells the function which data points to use. For example, if you want the slope of the Linear Regression line based on the Close price, replace PRICE with Close. The PRICE parameter can also be replaced by a numeric series input. You are not required to keep the PRICE parameter equal to a bar attribute such as Close. For example, if you wanted to calculate the linear regression of the Relative Strength Index, you can replace the PRICE parameter with RSI(Close,14).

The LEN parameter tells the function how many data points to consider in the regression. The input can be a number like 5, or you can replace it with a numeric simple type input. Numeric simple inputs limit you to simple expressions that return a number that doesn't change on a bar-to-bar basis. The value returned for the parameter LEN should be a positive whole number since it refers to the number of bars.

LinearRegValue

The LinearRegValue function projects, based on the current regression line, the regression values for x number of bars out into the future or x number of bars back in the past.

Category:

Easy Language

Function:

LinearRegValue(PRICE,LEN,TARGETB)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LEN the number of trailing bars to consider

TARGETB represents the number of bars into the future or back into the past.

Returns:

The current value of the specified regression line at TARGETB

Usage:

Linear Regression is a concept also known as the least squares method or best fit. Linear Regression attempts to fit a straight line between several data points in such a way that the distance between each data point and the line is minimized.

The equation of any line resembles the following:

$$y = mx + b$$

In the equation m refers to the slope of the regression line, b refers to the constant intercept of the y axis, x is the independent variable, and y is the dependent variable. This function returns the value y . The slope coefficient is the value returned by the function `LinearRegSlope`.

When the function is used in a study or system it requires three parameters from the user: `PRICE`, `LEN` and `TARGETB`.

The `PRICE` parameter tells the function which data points to use. For example, if you want the slope of Linear Regression line based on the `Close`, replace `PRICE` with `Close`. `PRICE` can also be replaced with a numeric series type input. You are not required to keep the parameter value equal to a bar attribute such as `Close`. For example, if you wanted to calculate the linear regression of the Relative Strength Index, you can replace the `PRICE` parameter with `RSI(Close,14)`.

The `LEN` parameter tells the function how many data points to consider in the regression. `LEN` can be hard coded with a number like 5 or be declared as a numeric simple input. The hard coded value or default value, if you choose to make `LEN` an input, can be any expression which uses simple mathematics like addition, subtraction, multiplication, or division. The value returned by the numeric simple type parameter cannot change on a bar by bar basis.

`TARGETB` represents the number of bars into the future or back into the past. If `TARGETB` is replaced by a positive number, the value returned by the `LinearRegValue` function will be for a bar in the past. If `TARGETB` is replaced by a negative number, the value returned by the `LinearRegValue` function will be for a bar in the future. For example, if you use a `TARGETB` of zero(0), the regression value returned would be for the current bar. However, if you use a -6, the value returned by the function would be the projected regression value for six bars out into the future. `TARGETB` can also be replaced with a numeric simple input.

Lowest

The `Lowest` function is designed to return the lowest `PRICE` over a period of time defined by the parameter `LENGTH`.

Category:

Easy Language

Function:

`Lowest(PRICE,LENGTH)`

Parameters:

`PRICE` specifies which price of the asset of interest is to be used

`LENGTH` the number of trailing bars to consider

Returns:

The lowest value in the period specified

Usage:

To understand the function, we must understand its parameters. The parameter `PRICE` is usually hard coded with some bar attribute such as `Close`, `Open`, `High`, `Low`, and `Volume`, or is replaced with a numeric series type input. However, it can be replaced with a valid Easy Language expression such as `Close + Open`, or `Average(RSI(Close,14),14)`.

The parameter `LENGTH`, just like `PRICE`, can be hard coded or can be replaced with a numeric simple type input. The value entered for `LENGTH` is usually whole positive number such as 5, 10, 14, etc. Once again, it can be replaced with a valid numeric expression. If you do decide to make it a numeric expression, keep in mind that value for `LENGTH` should be a positive whole number and cannot change on a bar-to-bar basis.

For the sake of simplicity, let's look at how the function works with `Close` for `PRICE` and 14 `LENGTH`. On each bar, the function returns the lowest `Close` over the previous trailing 14 bars. There may be times when two or more bars tie for the lowest value. When this happens, the function returns the `PRICE` for the most recent bar.

A common usage of the `Lowest` function is to find a breakout. For example, an important occurrence is when the value of the `PRICE` parameter for the current bar is the lowest value over the last x bars. When using the function in this way, you must remember to remove the current bar from the function.

For example, the statement `Close < Lowest(Close, 14)` will never be evaluated as true. Let's assume that the `Close` of the current bar was also the lowest `Close` over the last fourteen bars, this would only equate the two sides of the expression. To overcome this problem, you must exclude the last bar from the calculation of the function. This is done by offsetting the function by one bar. Thus, you will be comparing the `Close` of the current bar with the value

returned by the Lowest function for the previous bar. The correct statement is `Close<Lowest(Close,14)[1]`.

LowestBar

The LowestBar function is very similar to the Lowest function. The Lowest function returns the value of the lowest PRICE parameter over LENGTH bars, and the LowestBar function returns the number of bars ago that it occurred.

Category:

Easy Language

Function:

`LowestBar(PRICE,LENGTH)`

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH the number of trailing bars to consider

Returns:

The number of bars ago that the lowest value occurred

Usage:

The parameter PRICE, is usually hard coded with some bar attribute such as Close, Open, Low, Low, and Volume, or is replaced with a numeric series type input. However, can be replaced with a valid Easy Language expression such as `Close + Open`, or `Average(RSI(Close,14),14)`.

The parameter LENGTH, just like PRICE, can be hard coded or replaced with an numeric simple type input. The input is usually a positive whole number such as 5, 10, 14, etc. You may choose to replace LENGTH with a valid Easy Language expression. If you do decide to make it an expression, keep in mind that the value must be a whole number and cannot change on a bar-to-bar basis.

For the sake of simplicity, lets look at how the function works with Close for PRICE and 14 LENGTH. On each bar, the function returns the number of bars ago on which the lowest Close was achieved over the previous 14 trailing bars. In the event that two or more bars tie for the lowest value, the function identifies the most recent bar.

LRO

The LRO function returns the number of bars ago the specified expression was true. Or, if the specified expression did not occur within the last x number of bars, the function informs you of such.

Category:

Easy Language

Function:

`LRO(EXPR, LENGTH, OCCUR)`

Parameters:

EXPR true/false expression to check for (i.e., `Close > Open`)

LENGTH number of trailing bars to check

OCCUR which occurrence to check for (i.e., 1 = least recent, 2 = 2nd least recent, etc.)

Usage:

Many times, when describing the rules of an analysis technique, you need to identify when a certain condition occurred. The LRO function is specifically designed for this task. The functions power is in its ability to identify how long ago something occurred so that you may then use the information to access price data from that bar.

The function always returns a number representing how many bars ago the true/false expression was fulfilled (0 = current bar, 1 = one bar ago, 2 = 2 bars ago ...). If the function does not find a bar within the specified LENGTH which meets the criteria, it will return a -1.

When using the LRO function in a study or system, always look at the result of the LRO function before using the value in another calculation or function. If you use a -1 value as a bar number in another function, it will result in a Runtime error, which is related to the Maximum number of bars referenced by a study setting (also known as MaxBarsBack).

LWAccDis

The Larry Williams - Accumulation Distribution function is the cumulative total of a variable well call x. The variable x is then summed every bar. The variable x is calculated as follows:

If todays Close is greater than yesterdays Close then $x = \text{Close} - \text{TrueLow}$

If todays Close is less than yesterdays Close then $x = \text{Close} - \text{TrueHigh}$

If todays Close is equal to yesterdays Close then $x = 0$ (zero)

Category:

Easy Language

Function:

LWAccDis

Usage:

TrueLow is calculated as follows:

If yesterdays Close is less than todays Low then TrueLow = yesterdays Close

If yesterdays Close is greater than or equal to todays Low then TrueLow = todays Low

TrueHigh is calculated as follows:

If yesterdays Close is greater than todays High then TrueHigh = yesterdays Close

If yesterdays Close is less than or equal to todays High then TrueHigh = todays High

Signals are generated by examining divergences between the Williams Accumulation - Distribution and the underlying market. A new High in the market unaccompanied by a similar new High in the Williams Accumulation - Distribution is considered bearish. On the other hand, a new Low not accompanied by a new Low in the Williams Accumulation - Distribution is considered bullish.

MACD

The function returns the difference between a fast and slow moving average based on the same PRICE.

Category:

Easy Language

Function:

MACD(PRICE,FASTMA,SLOWMA)

Parameters:

PRICE specifies which price of the asset of interest is to be used

FASTMA number of trailing bars to consider for the fast average

SLOWMA number of trailing bars to consider for the slow average

Returns:

The MACD value for the current bar

Usage:

When the function is used in a study or system, PRICE, FASTMA, and SLOWMA can either be hard coded or replaced with a variable. The parameter PRICE is usually hard coded with some bar attribute such as Close, Open, Low, High, and Volume, or is replaced with a numeric series type input. However, it can be replaced with a valid Easy Language expression such as Close + Open or Average(RSI(Close,14),14).

FASTMA and SLOWMA refer to the number of bars used in the moving averages. Therefore, if hard coded, the value used to replace these two parameters should be a whole number and cannot change on a bar-by-bar basis. FASTMA is supposed to be less than SLOWMA. If the specified length of FASTMA is greater than that of the SLOWMA, the oscillator will invert.

During rising markets, the fast period moving average will rise faster than the slow period moving average resulting in a rising differential line or a larger value. The idea is that with a smaller number of bars used in its calculation, the fast period moving average, will more quickly indicate the trend of the most recent data.

When the fast period moving average crosses above or crosses below the slow period moving average, a signal is generated. Once the lines cross, it becomes important to look at the distance between them. The MACD function returns that difference. When the fast period moving average is above the slow period moving average, the MACD is positive and when it is below the slow period moving average the MACD is negative.

MassIndex

The MassIndex function is used to warn of an impending direction change. It uses two variables in its equation: Numerator and Denominator. The formula is:

MassIndex = Numerator/Denominator

...where

Numerator = Summation(XAverage(High-Low, LENGTH1), LENGTH2)

Denominator = XAverage(XAverage(High-Low, LENGTH1), LENGTH2)

Category:

Easy Language

Function:

MassIndex(LENGTH1, LENGTH2)

Parameters:

LENGTH1 number of periods for the exponential moving average

LENGTH2 number of periods for the summation and second exponential moving average
Usage:
The MassIndex is used to warn of an impending direction change. Therefore, the theory is that when a bulge occurs, you should take a position in the opposite direction.

Median

The Median function looks at a series of values and returns the median value. When the function is used in studies and systems, it requires two pieces of information from the user, PRICE and LENGTH.

Category:

Easy Language

Function:

Median(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of trailing bars to consider

Returns:

The median value for the period specified

Usage:

The parameter PRICE tells the function what values to look at, while the LENGTH parameter tells the function how many bars to include in the calculations. For example, if you hard coded the parameter PRICE with Close, and LENGTH with 4, the function would sort the last four closes in ascending order, choose the second and third largest values, and average them. See the table below.

Bar Number	Price	Rank (1 is the smallest)	Median
Close[0]	3646.62		
Close[1]	3625.01		
Close[2]	3667.13		
Close[3]	3669.64		

If LENGTH is an odd number, like 5, the function would sort the last five closes in ascending order, and choose the third (middle) value. See the table below.

Bar Number	Price	Rank (1 is the smallest)	Median
Close[0]	3646.62		
Close[1]	3625.01		
Close[2]	3667.13		
Close[3]	3669.64		
Close[4]	3670.05		

CLOSE and LENGTH can be hard coded or replaced with variables. PRICE is a numeric series type parameter. You can replace PRICE with a bar attribute, like High, or you can use any valid Easy Language expression such as RSI(Close, 14).

The parameter LENGTH, just like PRICE, can be hard coded or replaced with an numeric simple type input. The value is usually a positive whole number such as 5, 10, 14, etc. Once again, you can replace LENGTH with a valid Easy Language expression. If you do decide to make it an expression, keep in mind that LENGTH must be a whole number as it represents the number of bars, and it cannot change on a bar-to-bar basis.

MedianPrice

This function sums the High and Low of a bar and then divides that sum by two. The value returned by the function is the middle of the bar.

Category:

Easy Language

Function:

MedianPrice

Returns:

The value $(H+L)/2$

Usage:

Many times identifying simple bar patterns can lead to further development of more complex studies and systems.

MFI

The MFI function returns the calculated value of Range divided by Volume, or as follows:

$MFI = Range / Volume$

Category:

Easy Language

Function:

MFI(MYVOL)

Parameters:

MYVOL Total volume of current bar, at the end of the day

Usage:

Please refer to the discussion on the Range function in this chapter.

MidPoint

The MidPoint function finds the highest High and lowest High in the ten bar period, sums these two values, and then divides by two.

Category:

Easy Language

Function:

MidPoint(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH the number of trailing bars to consider

Returns:

The midpoint of the period specified

Usage:

When the function is used in a study or system, it requires the user to provide two pieces of information, PRICE and LENGTH. PRICE tells the function what data points to analyze and LENGTH tells the function how many bars to analyze. For example if you want to find the midpoint of the highs across ten bars at a time, you would replace PRICE and LENGTH with High and 10, respectively.

The parameter PRICE, is usually hard coded with some bar attribute such as Close, Open, Low, Low, and Volume, or is replaced with a numeric series type input. However, can be replaced with a valid Easy Language expression such as Close + Open or Average(RSI(Close,14),14).

The parameter LENGTH, just like PRICE, can be hard coded or replaced with an numeric simple type input. The value is usually positive whole number such as 5, 10, 14, etc. Once again, you may choose to replace LENGTH with a valid Easy Language expression. If you do decide to make it an expression, keep in mind that LENGTH must be a whole number and cannot change from bar to bar.

MinutesToTime

This convenient function, in conjunction with the TimeToMinutes function, can be used to take a time and add or subtract x number of minutes from that time.

Category:

Easy Language

Function:

MinutesToTime(MINUTES)

Parameters:

Minutes - number of minutes since midnight

Returns:

Time in military format (for example, 1530 = 3:30pm)

Usage:

Why do we need this function? We know that if the time was 3:30PM (15:30 in military time) and you wanted to add 50 minutes to the time, the new time would be 4:20PM. The computer, however, is not aware of the rules that we use to derive the new time. If you tell the computer to add 1530 + 50, it will return 1580 and as we know this is an invalid time.

Therefore, these functions have been built-in to provide the computer with the rules need to add and subtract time. The first step when adding or subtracting a number of minutes from time is to convert the time to minutes through the use of the TimeToMinutes function. Then add or subtract the number of minutes. Finally, convert the number of minutes back to time using the MinutesToTime function.

MINUTES can be hard coded with a number or can be replaced with a simple numeric type input.

Momentum

The Momentum study is an overbought/oversold oscillator. It is calculated by subtracting the value returned by the parameter PRICE for the current bar from the value returned by the parameter PRICE for the bar that occurred LENGTH bars ago. If the value for the current bar exceeds that of the bar in the past, the value of the function is positive. If the value for the current bar is less than that of the bar in the past, the value of the function is negative. Therefore, the values returned by the MOMENTUM function oscillates above and below zero.

Category:

Easy Language

Function:

Momentum(PRICE,LENGTH)

Parameters:

PRICE specifies some price of the asset of interest

LENGTH the number of trailing bars to consider

Returns:

Momentum value for the current bar

Usage:

If the market has decreased by more than x points, the market is considered to be oversold.

If the market has increased by more than x points, the market is considered to be overbought.

If you were to hard code the parameter PRICE with Close, and the parameter LENGTH with 10, the function would subtract the Close of 10 bars ago from the Close of the current bar.

MoneyFlow

The Money Flow function compares today's average price to yesterday's average price and then weighs the average price by the volume to calculate money flow (MF). The ratio of the summed positive and negative money flows are then normalized to be on a scale of zero to 100.

Category:

Easy Language

Function:

MoneyFlow(LENGTH)

Parameters:

LENGTH number of trailing bars used in calculation

Usage:

Average price is calculated by summing the Open (if available), Close, High, and Low and then dividing the sum by either four or three (three if no Open is available). Positive money flow occurs when today's average price exceeds yesterday's average price, and is calculated by multiplying today's volume by today's average price. Negative money flow occurs when yesterday's average price exceeds today's average price, and is calculated by multiplying today's volume by today's average price. The positive and negative money flows are then summed over the number of bars specified in LENGTH. The Money Flow Index is then calculated as follows:

$$MFI = 100 - (100 / (1 + (\text{Sum of Positive MF} / \text{Sum of Negative MF})))$$

MoneyFlow issues a signal when a new High or Low is reached in the market which is not confirmed by a similar new high in the Money Flow Index.

MorningStar

The MorningStar function determines an average length of the body and evaluates the current and the two previous bars. The function compares the body of the bar preceeding the previous bar to the average body, and then looks at the position of the previous and the current bar to determine whether or not the current bar completes a MorningStar candlestick formation.

Category:

Easy Language

Function:

MorningStar(Length)

Parameters:

Length the number of trailing bars used in determining the average length of the body.

Returns:

True/False

MRO

Many times, when describing the rules of an analysis technique, you need to identify when a certain condition occurred. The MRO function is specifically designed for this task; the functions power lies in its ability to identify how long ago something occurred and then use that information to access price data from those bars.

Category:

Easy Language

Function:

MRO(EXPR,LENGTH,OCCUR)

Parameters:

EXPR occurrence to check for, for example when Close>Open

LENGTH number of trailing bars to check

OCCUR which occurrence, for example, 1 = most recent, 2 = 2nd most recent, and so on

Returns:

Number of bars ago specified expression was true; -1 if expression was not found to be true within the last LENGTH bars

Usage:

The first parameter, EXPR, stands for expression. It is the true/false condition you are looking for. For example, if you wanted to find the most recent occurrence of an outside bar, that would be your expression. LENGTH refers to how many bars the MRO function will evaluate at a time. Find the most recent occurrence of an outside bar for every 15 bars would require a LENGTH of 15. OCCUR stands for occurrence and refers to which occurrence you want to find. Usually the most recent occurrence is desired and is obtained by using an OCCUR of 1. If you wanted to find the second most recent occurrence, you would use the number 2.

Since the parameter EXPR is a true/false parameter, its replacement must be a true/false expression. The hard coded value or the default value of the input used to replace it must be a true/false expression. The parameters LENGTH and OCCUR are numeric simple type parameters and can be replaced with any positive whole number or numeric expression that yield a whole number. And, the value that replaces OCCUR should not be greater than the value that replaces LENGTH.

The MRO function always returns a number representing how many bars ago the true/false expression occurred (0 equals the CurrentBar, 1 equals one bar ago, etc.). If it did not occur within the LENGTH of bars specified, the MRO function returns -1.

Note: Test the return value before using it as a bar reference. If the MRO function does not find your specified criteria (and therefore returns a -1), normally you would not want to take any action. Therefore, check to see if the function returns a value greater than negative one. If you try to reference a price using a bar number of -1 you will receive a Runtime error, which refers to the Maximum number of bars referenced by a study (or MaxBarsBack) setting.

MyPrice

The MyPrice function returns the average price of the bar. It sums the High, Low and Close prices of the bar and divides the sum by 3.

Category:

Easy Language

Function:

MyPrice

Next3rdFriday

The Next3rdFriday function returns the number of days to the next third Friday in the month.

Category:

Easy Language

Function:

Next3rdFriday(SERIES)

Parameters:

SERIES Option series for which to calculate the number of days to the next 3rd Friday

NthHighest

The Highest function is perfect for finding the highest value returned by the parameter PRICE for the number of bars specified by the parameter LENGTH. However, there may be times when you want to find the second or third highest value. The NthHighest function was specifically designed for this purpose.

Category:

Easy Language

Function:

NthHighest(NTH,PRICE,LENGTH)

Parameters:

NTH which element (i.e., 1 = highest, 2 = 2nd highest, and so on)

PRICE specifies which price of the asset of interest to be used

LENGTH the number of trailing bars to consider

Returns:

Value of Nth highest occurrence

Usage:

Just like the Highest function, the NthHighest function has the parameters PRICE and LENGTH. In addition, it also has the parameter NTH. This is the parameter that specifies if its the second, third, etc. highest value desired. For example, if you wanted the second highest Close for the last five bars, you would replace NTH with 2, PRICE with Close and LENGTH with 5.

Note: The LENGTH parameter in the NthHighest function can only be replaced with a value equal to or less than 25. The value that replaces NTH should not be less than the value that replaces the parameter LENGTH.

NthHighestBar

The HighestBar function is perfect for finding the highest value returned by the parameter PRICE for the specified LENGTH. However, there may be times when you want to find the second or third highest bar. The NthHighestBar function was specifically designed for this purpose.

Category:

Easy Language

Function:

NthHighestBar(NTH,PRICE,LENGTH)

Parameters:

NTH specifies which element (i.e., 1 = highest, 2 = 2nd highest, and so on)

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of trailing bars to consider

Returns:

Number of bars ago the Nth highest value occurred

Usage:

Just like the HighestBar function, the NthHighestBar function has the parameters PRICE and LENGTH. In addition, it also has the parameter NTH. This is the parameter that specifies whether its the second, third, etc. highest bar desired. For example, if you wanted to know how many bars ago the second highest Close for the last five bars took place, replace NTH with 2, PRICE with Close and LENGTH with 5.

Note: The LENGTH parameter in the NthHighestBar function can only be replaced with a value equal to or less than 25. The value that replaces NTH should not be less than the value that replaces the parameter LENGTH.

NthLowest

The Lowest function is perfect for finding the lowest value returned by the parameter PRICE for the specified LENGTH. However, there may be times when you want to find the second or third lowest value. The NthLowest function was specifically designed for this purpose.

Category:

Easy Language

Function:

NthLowest(NTH,PRICE,LENGTH)

Parameters:

NTH specifies which element (i.e., 1 = lowest, 2 = 2nd lowest, and so on)

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of trailing bars to consider

Returns:

Value of Nth lowest occurrence

Usage:

Just like the Lowest function the NthLowest function has the parameters PRICE and LENGTH. In addition, it also has the parameter NTH. This is the parameter that specifies whether its the second, third, etc. lowest value desired. For example if you wanted the second lowest Close for the last five bars, replace NTH with 2, PRICE with Close and LENGTH with 5.

Note: The LENGTH parameter in the NthLowest function can only be replaced with a value equal to or less than 25. The value that replaces NTH should not be greater than the value that replaces the parameter LENGTH.

NthLowestBar

The LowestBar function is great at being able to find the lowest PRICE for the specified LENGTH. However, there may be times when you want to find the second or third lowest bar. The NthLowestBar function was specifically designed for this purpose.

Category:

Easy Language

Function:

NthLowestBar(NTH,PRICE,LENGTH)

Parameters:

NTH specifies which element (i.e., 1 = lowest, 2 = 2nd lowest, and so on)

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of trailing bars to consider

Returns:

Number of bars ago the Nth lowest value occurred

Usage:

Just like the LowestBar function, the NthLowestBar function has the parameters PRICE and LENGTH. It also has the parameter NTH. This is the parameter that specifies whether its the second, third, etc. lowest bar desired. For example, you would use this function when you wanted to know how many bars ago there occurred the second lowest Close.

Note: The LENGTH parameter in the NthLowestBar function can only be replaced with a value equal to or less than 25. The value that replaces NTH should not be less the value that replaces the parameter LENGTH.

OBV

The On Balance Volume function attempts to gauge the buying and selling pressure in the market.

Category:

Easy Language

Function:

OBV

Returns:

The current OBV value

Usage:

Mathematically, the cumulative OBV formula is represented as:

$OBV = [(C - C_p) / |C - C_p|] * V$

... where

C is the current periods closing price.

C_p is the previous periods closing price.

|C - C_p| is the absolute value of the difference between the two closing prices.

Since any number divided by itself is equal to one (1), the only bearing the expression in parenthesis has is on the sign. If the previous Close is greater than the current Close, the sign will be negative. If the previous Close is less than the current Close, the sign will be positive.

Therefore, the function either subtracts the current volume from the running total of the OBV line or adds current volume to the running total of the OBV line.

There are several ways to analyze OBV, including variations of trend assessment and divergence analysis. The function has no inputs as it is totally based on the volume of the bar.

Parabolic

The Parabolic system, as described in technical analyst Welles Wilders book New Concepts In Technical Trading Systems, is a complete stop-setting entry and exit trading system. The system is designed to allow more leeway or tolerance for contra-trend price fluctuation early in a new trade, then to progressively tighten a protective trailing stop order as the trend matures.

Category:

Easy Language

Function:

Parabolic

Parameters:

AF specifies the Acceleration Factor

Returns:

The current Parabolic SAR value

Usage:

The concept draws on the idea that time is an enemy, and unless a trade can continue to generate more profits over time it should be liquidated. Thus, the Parabolic Time/Price System rides the trend until the SAR price is penetrated. Then the existing position is closed out and the reverse position is opened.

The expression parabolic derives from the shape of the curve the stops create as they appear on the chart (this can be seen when the Parabolic indicator is applied to a data series on a chart window).

To calculate the function, we must first find an extreme reference point. On the long side this price is usually the lowest price recorded during the previous closed short position. On the short side this extreme price is usually the highest price recorded during the previous closed out long position.

In practice, however, you won't have an extreme reference point for the very first trade as there are no previous trades. To account for this, the function uses, as a default, either the High or Low of the previous bar before the first trade.

For long-side sells and short-side buy, on the second day and thereafter, the SAR is adjusted as follows:

$SAR_b = SAR_p + AF(H - SAR_p)$

$SAR_s = SAR_p + AF(L - SAR_p)$

...where

SAR_b is the long-side sell stop price at which you sell and sell short.

SAR_s is the short-side buy stop at which you buy and go long.

SAR_p is the previous bars SAR

AF is an acceleration factor that begins at .02 for the bar after the initial SAR buy stop order opens the long trade and is increased by .02 each period that price rises to the highest level (H) since the current long trade was opened. For periods when price does not set a new High within the current long trade time duration, AF is left unchanged from its previous periods level.

H is a new highest price since the current long trade was opened on a stop buy order.

L is a new lowest price since the current short trade was opened on a stop sell order.

The value returned by the Parabolic function is the SAR value described above.

PercentR

Percentage Range is essentially a technique for determining oversold and overbought signals in trading markets. The study uses the price range over a given period of time to establish a normal trading range. It then identifies occurrences of prices outside this normal trading range.

Category:

Easy Language

Function:

PercentR(LENGTH)

Parameters:

LENGTH specifies the number of trailing bars to consider

Returns:

The current %R value

Usage:

The Percent Range study was first introduced by Larry Williams in his book called, How I Made One Million Dollars Last Year Trading Commodities (Monterey, California: Conceptual Management, 1973 2nd edition).

Percentage R is most effective in trading markets with defined cycles and in trending markets without a reputation for cycles. Traders should not use %R by itself since it will give false signals in trending markets. It is recommended that trend evaluating studies such as moving averages or Elliot wave be used simultaneously to confirm the signals generated by the %R study.

Williams Percent R differs from most overbought/oversold studies in that the higher the value of the study the more oversold the market and the lower the value of the study the more overbought the market. To be consistent with the other oscillators, Omega Research takes Williams value and subtracts it from one hundred (100). For example, if Williams Percent R returns 80, Omega's Percent R will return 20.

The formula used to calculate the Percentage Range follows:

$$\%R = 100 * ((Hr - C) / (Hr - Lr))$$

...where

r = The time period selected

Hr = The highest High of that period

Lr = The lowest Low of that period

C = Today's Close

The period should be at least five trading days, and half the length of an identifiable market cycle. The more volatile a market, the shorter traders will want to make the period they use for their range. The period will vary with each trader's market time frame. For a market of average volatility, a good period would be ten days.

The parameter LENGTH can be hard coded with a whole number or can be replaced by a numeric simple input whose default value is a whole number.

PriceVolTrend

The Price Volume Trend (PVT) function multiplies the day's trade volume by the percentage difference between today's close and yesterday's close. It accumulates the resulting value for each day, enables you to view the resulting trend, either up or down.

Category:

Easy Language

Function:

PriceVolTrend

Usage:

The function is calculated as follows:

$$\text{PriceVolTrend} = (((C - C[1]) / C[1]) * V)$$

... for the first day, and as follows for subsequent days:

$$\text{PriceVolTrend} = (((C - C[1]) / C[1]) * V) + \text{PriceVolTrend}[1];$$

Put

The Put function can be used as a substitute or input for the parameter TYPE in many of the Option indicators (i.e., Black Scholes, OptionProfile, etc.). The TYPE parameter must be replaced by a 0 or a 1, to indicate either a Put or a Call. Using this function or the Call function eliminates the need to remember the correct number to enter. Instead, you can just enter the word Put or Call.

Category:

Easy Language

Function:

Put

Usage:

This function always returns a 0, to indicate a Put.

Range

The function performs the task of subtracting the Low of a bar from the High.

Category:

Easy Language

Function:

Range

Returns:

The current (H-L) value

Usage:

This study can help identify periods where volatility may be about to pick up.

RangeLeader

The RangeLeader function compares the current and previous bar, and looks at two conditions: 1) whether the mid-point of the current bar is greater than the previous High or less than the previous Low, and 2) whether the Range of the current bar is greater than the Range of the previous bar. If both conditions are met, the function returns a 1; if one or neither is met, the function returns a 0.

Category:

Easy Language

Function:

RangeLeader

Usage:

The formula or code for this function is as follows:

Value1 = (High + Low) / 2 ;

Cond1 = Value1 > High[1] OR Value1 < Low[1] ;

Cond2 = Range[0] > Range[1] ;

if Cond1 and Cond2 then

RangeLeader = 1

else

RangeLeader = 0 ;

RateOfChange

The Rate of Change (ROC) indicator is one of the easiest and most effective indicators you'll find. The ROC indicator lends itself handsomely to overbought and oversold interpretation. The problem is that there are no hard-and-fast rules about where the lines should be drawn (even more reason to make them variables), since the magnitude of the oscillations will vary according to the volatility of the underlying security and the time span being considered. For this reason overbought and oversold lines are constructed on the basis of judgment.

Category:

Easy Language

Function:

RateOfChange(Price,Length)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of trailing bars to consider

Returns:

The current ROC value

Usage:

Analysis should not be limited to oscillators based on one time span but should include other periods in order to obtain a more complete picture of what is happening. For example, a trader may be using a 5 day ROC which moves easily between overbought and oversold extremes. However, as this 5 day span indicates an oversold region the other short term time spans does not confirm. This is why it is often a good idea to monitor several different ROC indicators simultaneously.

A word of caution, the rate of change must be judged as too erratic relative to some other indicators, such as moving averages. The reason for this seems to be because the equation for ROC has an over dependence on the oldest data.

The ROC indicator can be calculated in one of two ways: the subtraction method or the division method. The Omega ROC function uses the division method. The equivalent of the subtraction method is found in the Momentum function.

$$ROC = ((Price/Pricep) - 1) * 100$$

...where

Price = Current value (i.e., Close, High, Low, etc.)

Pricep = Previous value (determined by the value returned for the input LENGTH)

This formula is somewhat different from what you may see in some publications that do not subtract one. From an interpretive point of view it is immaterial which method of scaling is used because the general movements are identical. Omega Research prefers to use positive and negative numbers rather than strictly positive percentages, since this gives a better sense of bullish and bearish tendencies.

The parameter PRICE, is usually hard coded with some bar attribute such as Close, Open, High, Low, and Volume or is replaced with a numeric series type input. However, it can be replaced with a valid Easy Language expression. For example, Close + Open, or Average(RSI(Close,14),14).

The parameter LENGTH, just like the parameter PRICE, can be hard coded or can be replaced with an numeric simple type input. The value hard coded or entered as a default value for the input is usually number such as 5, 10, 14, etc. Once again, you may choose to replace LENGTH with a valid numeric expression. If you do decide to make it a numeric expression there is one thing to keep in mind. The value returned for the parameter LENGTH should be a whole number cannot change on a bar-to-bar basis.

RSI

The Relative Strength Index (RSI) is a study introduced by technical analyst Welles Wilder in his book New Concepts in Technical Trading Systems published in 1978. Wilder cites three reasons for using RSI:

1. to avoid the erratically High values returned by other oscillator-type indicators such as Momentum or Rate of Change. Wilder sought some way to dampen or smooth out the extreme points used to calculate the oscillator.
2. it always returns a value between zero and one hundred, regardless of the asset it is applied to. This allows one to identify the most active commodities by identifying those in which the RSI is showing the greatest vertical movement - either up or down.
3. it is easy to calculate and keep track of. After calculating the initial RSI, only the previous days data is required for the next calculation.

Category:

Easy Language

Function:

RSI(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of trailing bars to consider

Returns:

The current RSI value

Usage:

The equation for the Relative Strength Index, RSI is:

$$RSI = 100 - (100 / (1 + RS))$$

$$RS = \text{Average of 14 days closes UP} / \text{Average of 14 days closes DOWN}$$

To calculate the first RSI value:

1. Calculate the sum of the UP closes for the previous 14 days and divide this sum by 14. This is the average UP Close.
2. Calculate the sum of the DOWN closes for the previous 14 days and divide by 14. This is the average DOWN Close.
3. Divide the average UP Close by the average DOWN Close. This is the Relative Strength (RS).
4. Add 1.00 to the RS.
5. Divide the result obtained in Step 4 into 100.
6. Subtract the result obtained in Step 5 from 100. This is the first RSI.

To calculate the RSI each time following the first RSI value:

1. To obtain the next average UP Close: Multiply the previous average Up Close by 13, add to this amount today's UP Close (if any) and divide the total by 14.
2. To obtain the next average DOWN: Multiply the previous average DOWN Close by 13, add to this amount today's DOWN Close (if any) and divide the total by 14.
3. Steps (3), (4), (5) and (6) are the same as for the initial RSI.

Wilder then goes on to point out five possible uses for the RSI:

The first usage is with tops and bottoms. The Index will usually top out or bottom out before the actual market top or bottom, giving an indication that a reversal or at least a significant reaction is imminent.

The second usage is with chart formations. The RSI line will sometimes form patterns such as head and shoulders tops or bottoms, pennants or triangles which indicate breakouts and buy and sell points.

The third usage is called Failure Swings. These swings occur when the RSI tops out or bottoms out, retraces slightly to what is known as the fail point, shows some indecision (sideways movement) and then retraces past the fail point.

The fourth usage is Support and Resistance. Areas of support and resistance often show up clearly on the index before becoming apparent on the bar chart.

The fifth and final usage Wilder points out is Divergence. Divergence is a phenomenon where the RSI is moving in the opposite direction of price.

Although Wilder uses fourteen for the period and Close for the price in his work, Omega Research gives you the flexibility to modify these parameters. The parameter PRICE is usually hard coded with some bar attribute such as Close, Open, High, Low, and Volume or is replaced with a numeric series type input. However, it can be replaced with a valid Easy Language expression, for example, Close + Open or Average(RSI(Close,14),14). The parameter LENGTH, just like PRICE, can be hard coded or replaced with an numeric simple type input. The value hard coded or entered as a default value for the input is usually number such as 5, 10, 14 etc. Once again, you may choose to replace LENGTH with a valid numeric expression. If you do decide to make it a numeric expression, keep in mind that the value returned for the parameter LENGTH should be a whole number and cannot change on

a bar-to-bar basis. An illustration of these concepts is shown in the figure below.
Example of failure swing usage of RSI

ShootingStar

The ShootingStar function evaluates the current bar and determines whether or not the candlestick formation is a ShootingStar formation.

Category:

Easy Language

Function:

ShootingStar(Tail)

Parameters:

Tail the length of the body (the distance from the open to the close) is multiplied by this number to determine the minimum acceptable length of the tail (the distance from the low to the body).

Returns:

True/False

SlowD

This is a stochastic function, smoothed b use of a moving average technique. Stochastics are oscillators that are used to indicate overbought and oversold conditions in the market based on the premise that during periods of price decreases, bar closes tend to accumulate near the Low of the bar, and during periods of price increases, bar closes tend to accumulate near the High of the bar.

Category:

Easy Language

Function:

SlowD(LENGTH)

Parameters:

LENGTH indicates number of bars used in the calculation

Usage:

Please refer to the discussion under the Stochastics function.

SlowDCustom

This is a stochastic function, which is smoothed by use of a moving average technique. Stochastics are oscillators that are used to indicate overbought and oversold conditions in the market based on the premise that during periods of price decreases, bar closes tend to accumulate near the Low of the bar, and during periods of price increases, bar closes tend to accumulate near the High of the bar.

Category:

Easy Language

Function:

SlowDCustom(HPRICE, LPRICE, CPRICE, LENGTH)

Parameters:

HPRICE High price to be used in calculation
LPRICE Low price to be used in calculation
CPRICE Closing price to be used in calculation
LENGTH indicates number of bars used in the calculation
Usage:
Please refer to the discussion under the Stochastics function.

SlowK

This is a stochastic function, which has been smoothed by use of a moving average technique. Stochastics are oscillators that are used to indicate overbought and oversold conditions in the market based on the premise that during periods of price decreases, bar closes tend to accumulate near the Low of the bar, and during periods of price increases, bar closes tend to accumulate near the High of the bar.

Category:

Easy Language

Function:

SlowK(LENGTH)

Parameters:

LENGTH indicates number of bars used in the calculation

Usage:

Please refer to the discussion under the Stochastics function.

SlowKCustom

This is a stochastic function, which is smoothed by use of a moving average technique. Stochastics are oscillators that are used to indicate overbought and oversold conditions in the market based on the premise that during periods of price decreases, bar closes tend to accumulate near the Low of the bar, and during periods of price increases, bar closes tend to accumulate near the High of the bar.

Category:

Easy Language

Function:

SlowKCustom(HPRICE, LPRICE, CPRICE, LENGTH)

Parameters:

HPRICE High price to be used in calculation

LPRICE Low price to be used in calculation

CPRICE Closing price to be used in calculation

LENGTH indicates number of bars used in the calculation

Usage:

Please refer to the discussion under the Stochastics function.

SmoothedAverage

The Smoothed Average function further smoothes an average of the last x bars. It does this by using the previous value of itself. Use the function in the same way you would use the Average function.

Category:

Easy Language

Function:

SmoothedAverage(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of trailing bars to consider

Returns:

The current Smoothed Average value

Usage:

On the first bar, this function adds together all the values returned by the parameter PRICE for the specified LENGTH, divides the sum by the LENGTH, then stores the value in a variable called SUM. Only on the first bar is the SmoothedAverage function equal to the value stored in SUM.

SUM = Summation(PRICE, LENGTH)

On each bar there after, the function uses a different equation.

SmoothedAverage = (Sum[1] - SmoothAverage[1] + Price)/Length

StdDev

This function returns a value that represents how widely dispersed the individual PRICE values are away from the mean average (using the same parameters).

Category:

Easy Language

Function:

StdDev(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of trailing bars to consider

Returns:

The current Standard Deviation

Usage:

The standard deviation is derived by first finding the variance and then taking its square

root:

...where

N is the number of elements

D is the individual elements in the population

M is the population mean

Stochastics

The stochastic oscillators indicate overbought and oversold areas in the market, based upon momentum or price velocity.

Category:

Easy Language

Function:

FastK(LENGTH),FastD(LENGTH),SlowK(LENGTH),SlowD(LENGTH)

Parameters:

LENGTH specifies the number of trailing bars to consider

Returns:

The current value of the specified stochastic function

Usage:

Stochastics are made up of four individual calculations: FastK, FastD, SlowK, SlowD. The core of the calculations are based upon the FastK value which is calculated in the following manner.

$$\text{FastK}(\text{Length}) = (C - L) / (H - L) * 100$$

... where

L=Lowest Low of a specified period

H=Highest High of a specified period

C=Today's Close

Length=Specified period of time

FastD is an exponentially smoothed FastK using a .5 factor in the smoothing. Both the FastK and FastD are used in the Stochastic-Fast indicator. The slow stochastic is created with the SlowK which equals the FastD. The SlowD is a three period smoothed SlowK.

When using stochastics, most people use the slow stochastic since its smoothed values are less subject to whipsaws. The indicator itself is composed of a percentK and a percentD line that will always oscillate between 0 and 100; a higher value indicates an overbought market while a lower value indicates an oversold market.

A value greater than 80 is usually used as an overbought signal and a value less than 20 is used for the oversold signal. The 80 value is often called the sell zone and the 20 value is called the buy zone. The crossover of the buy zone and sell zone values in conjunction with a percentD and percentK crossover is a good indication of a switch in trend direction.

George C. Lane, developer of the stochastic, says the only valid signal derived from the percentD is a divergence from the price where the divergence is an indication of a trend reversal. When using the percentK and the percentD together, crossovers of the two generate valid signals and should be used to indicate rising and declining markets.

Summation

This function performs a very basic but useful task. It adds a series of numbers together.

Category:

Easy Language

Function:

Summation(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of trailing bars to consider

Returns:

The sum of the last LENGTH occurrences of PRICE

Usage:

The number of values in the series is determined by the parameter LENGTH. The values added together are determined by the parameter PRICE. For example, if the parameters PRICE and LENGTH were replaced by Close and 14, respectively, the function would add together the last fourteen closes.

Note: The current bars PRICE is included in the calculation, unless the function is offset.

SwingHigh

The SwingHigh function returns the PRICE of the Swing High bar. A Swing High occurs when the PRICE of a bar is higher than the PRICE on the preceding bar(s), and at least as high as the same PRICE on the bar(s) that follow it.

An illustration of this concept is shown below in the figure below.

An illustration of a Swing High Bar

If you were to replace the parameter PRICE with the word High, the High of the Swing High bar would be returned by the SwingHigh function.

Category:

Easy Language

Function:

SwingHigh(OCCUR,PRICE,STRENGTH,LENGTH)

Parameters:

OCCUR which occurrence (i.e., 1 = most recent, 2 = 2nd most recent, and so on)

PRICE specifies which price of the asset of interest is to be used

STRENGTH specifies required number of bars on either side of swing bar

LENGTH specifies the number of trailing bars to consider

Returns:

The specified SwingHigh value or -1 if not found

Usage:

The previous figure shows a Swing High based on the Highs of the bars. STRENGTH is the number of bars on each side of the Swing High. A strength of one indicates that the value returned by the parameter PRICE must be greater than the same value returned for the bar

on its left and equal to or greater than the bar on its right. The figure below shows a Swing High with a STRENGTH of two.

Swing High Bar with a strength of two

In both of the figures, the parameter PRICE has been substituted with the value High. The parameter LENGTH refers to the number of bars being examined for the Swing High. The

parameter OCCUR refers to which Swing High you want to use. For example, if in a twenty-one bar period three swing highs were found, it becomes necessary to specify which Swing High is desired. If the most recent Swing High is desired, a one (1) would be substituted for the parameter OCCUR.

Note: If no Swing High is found in the period (LENGTH) specified, the function will return a minus one (-1). The value of the parameter LENGTH must exceed STRENGTH by at least one. In addition, the Maximum number of bars referenced by a study (known as MaxBarsBack) must be greater than the sum of the values of STRENGTH and LENGTH.

SwingHighBar

This function works in the same way the SwingHigh function does with one exception. Instead of returning the PRICE of the bar, the function returns how many bars ago the Swing High occurred.

Category:

Easy Language

Function:

SwingHighBar(OCCUR,PRICE,STRENGTH,LENGTH)

Parameters:

OCCUR which occurrence (i.e., 1 = most recent, 2 = 2nd most recent, and so on)

PRICE specifies which price of the asset of interest is to be used

STRENGTH specifies required number of bars on either side of the swing bar

LENGTH specifies the number of trailing bars to consider

Returns:

The number of bars ago the specified Swing High occurred, or -1 if not found

Usage:

Please refer to the SwingHigh function for more information on the usage of SwingHighBar.

SwingIndex

Technical analyst Welles Wilder created the Swing Index because he was in search of a way to cut through the maze of Open, High, Low and Close prices to indicate the real strength and direction of the market. The Swing Index looks at the Open, High, Low, and Close values between a two bar period.

Category:

Easy Language

Function:

SwingIndex

Returns:

The current Swing Index value

Usage:

Wilder points out that there are four cross-bar comparisons and one intra-bar comparison that are strong indicators of an UP day or a DOWN day:

1. Close today above or below previous Close.
2. Close today above or below Open today.
3. High today above or below previous Close.
4. Low today above or below previous Close.
5. Previous Close minus previous Open.

The Swing Index returns a number between positive and negative one hundred. If the factors pointed toward an UP day the function value would be positive. If the factors pointed toward a DOWN day the function value would be negative.

...where

K = the larger of

1. H2-C1
2. L2-C1

L = value of a limit move in one direction.

To obtain R, first determine the largest of:

1. H2-C1

2. L2-C1

3. H2-L2

If (1) is the largest, $R=(H2-C1)-.5(L2-C1)+.25(C3-O1)$

If (2) is the largest, $R=(L2-C1)-.5(H2-C1)+.25(C3-O1)$

If (3) is the largest, $R=(H2-L2)+.25(C3-O1)$

Let us evaluate the contribution the Swing Index can make. The Swing Index gives us definite short-term swing points. It can be used to supplement other methods as a breakout indicator. A breakout is indicated when the value of the Accumulation Swing Index (ASI) exceeds the ASI value on the day when a previous significant high SWING POINT was made. A downside breakout would be indicated when the value of the ASI drops below the ASI value on a day when a previous significant LOW SWING POINT was made.

The ASI is obtained by accumulating each days Swing Index(SI) as indicated by the sign (+ or -) of the latest SI. ASI may be either minus or plus. If the long-term trend is up, ASI will be plus. If the long-term trend is down, ASI will be a minus.

Finally, since the SI always returns a number between +/- 100, it can be compared across different securities that have the same daily limit to rank the strength of the swings.

Note: Since only futures have a relative daily limit value, this function only makes sense if it is applied to a futures contract. If you use the Swing Index Indicator or Accumulation Swing Index Indicator and it only plots a zero flat line check the Daily Limit (Chart Setup - Settings Command menu sequence) value.

SwingLow

The SwingLow function returns the PRICE of the Swing Low bar. A Swing Low occurs when the PRICE of a bar is less than the same PRICE on the preceding bar(s), and at least as low as the same PRICE on the bar(s) that follow it. This concept is illustrated in the figure below.

An illustration of a Swing Low Bar

If you were to replace the parameter PRICE with the word Low, the Low of the Swing Low bar would be returned by the SwingLow function.

Category:

Easy Language

Function:

SwingLow(OCCUR,PRICE,STRENGTH,LENGTH)

Parameters:

OCCUR which occurrence (i.e., 1 = most recent, 2 = 2nd most recent, and so on)

PRICE specifies which price of the asset of interest is to be used

STRENGTH specifies required number of bars on either side of swing bar

LENGTH specifies the number of trailing bars to consider

Returns:

The specified Swing Low value, or -1 if not found

Usage:

The figure above shows a Swing High based on the lows of the bars. STRENGTH is the number of bars on each side of the Swing Low. A strength of one indicates that the value returned by the parameter PRICE must be less than the same value returned for the bar on its left, and equal to or less the bar on its right.

The figure above shows a Swing Low based on the lows of the bars. Since only one bar is shown on either side this Swing Low is said to have a STRENGTH of one. Figure 6-64 shows a Swing Low with a STRENGTH of two. The larger the STRENGTH, the more major the valley.

In both of the figures, the parameter PRICE has been substituted with the value Low. The

parameter LENGTH refers to the number of bars being examined to find the Swing Low. The parameter OCCUR refers which Swing Low you want to use. For example, if in a twenty-one bar period three Swing Lows were found, it becomes necessary to specify which Swing Low is desired. If the most recent Swing Low is desired a one would be substituted for the parameter OCCUR.

Note: If no Swing Low is found in the period (LENGTH) specified, the function will return a minus one (-1). The value returned by the parameter LENGTH must exceed the value returned by the parameter STRENGTH by at least one. In addition, the Maximum number of bars referenced by a study setting (known as MaxBarsBack) must be greater than the sum of the values of STRENGTH and LENGTH.

SwingLowBar

This function works in the same way the SwingLow function does with one exception. Instead of returning the PRICE of the bar, the function returns how many bars ago the Swing Low occurred.

Category:

Easy Language

Function:

SwingLowBar(OCCUR,PRICE,STRENGTH,LENGTH)

Parameters:

OCCUR which occurrence (i.e., 1 = most recent, 2 = 2nd most recent, and so on)

PRICE specifies which price of the asset of interest is to be used

STRENGTH specifies required number of bars on either side of the swing bar

LENGTH specifies the number of trailing bars to consider

Returns:

The number of bars ago specified Swing Low value occurred, or -1 if not found

Usage:

For more information, please refer to the description under the SwingLow function.

Theta

This function measures the amount by which an option price declines due to the passage of 1 day.

Category:

Easy Language

Function:

Theta(DAYSLEFT, STRIKE, LEVEL, RATE, VOLA, TYPE)

Parameters:

DAYSLEFT number of days left to the expiration of the option

STRIKE exercise price of the option

LEVEL price of the underlying asset

RATE currently available risk-free interest rate

VOLA annual volatility of the underlying asset

TYPE Put = 0; Call = 1

Usage:

An Option derives some portion of its value from the time remaining until it expires. The longer this time is, the more the option is worth, all else being equal.

TimeSeriesForecast

The Time Series Forecast function can be used to plot a line through the prices of a security in an attempt to minimize the distance between the line and each individual point. It then extends that line BARPLUS bars into the future.

Category:

Easy Language

Function:

TimeSeriesForecast(LENGTH, BARSPLUS)

Parameters:

LENGTH the number of trailing bars to consider

BARPLUS number of bars into the future the linear regression line is to be extended

Usage:

The method used to do this is called the Least Squares method. The TimeSeriesForecast formula is:

TimeSeriesForecast = LinearRegValue(Close, LENGTH, BARSPLUS)

The Time Series Forecast function is based on the theory that there is an underlying force that will pull prices back to the regression line when they stray above or below it. When the price is above the projected regression line, it is an indication that the trend is up. However, when it is below the projected regression line, it is an indication that the trend is down.

TimeToMinutes

This convenient function, in conjunction with the MinutesToTime function, can be used to take a time and add or subtract x number of minutes from that time.

Category:

Easy Language

Function:

TimeToMinutes(XTIME)

Parameters:

XTIME time, in the 24-hour format

Returns:

The number of minutes between XTIME and the preceding midnight

Usage:

Why do we need this function? We know that if it is 3:30pm (15:30 in military time) and you want to add 50 minutes, the new time would be 4:20pm. However, if you tell the computer to add 3:30 + 50, it will return 3:80, which is meaningless.

Therefore, these functions have been built-in to supply the computer with the rules needed to properly add and subtract time. The first step when adding or subtracting a number of minutes from time is to convert the time to minutes through the use of the TimeToMinutes function.

Then, add or subtract the number of minutes, and, finally convert the number of minutes back to time using the MinutesToTime function.

MINUTES is a numeric simple type parameter and can be hard coded with a number or can be replaced by a numeric simple input.

TAngleEasy

The TAngleEasy function returns the angle of a trendline, if that trendline were to be drawn between two data points on your chart.

Category:

Easy Language

Function:

TAngleEasy(PRICE, LEFTBAR, RIGHTBAR)

Parameters:

PRICE any bar attribute (i.e., Close, Open, etc.) or any formula

LEFTBAR refers to the bar the furthest back in time

RIGHTBAR refers to the most recent bar on your chart

Usage:

Use positive numbers to project back in time and use negative numbers to project into the future.

TSlopeEasy

The TSlopeEasy function returns the slope of a trendline, if that trendline were to be drawn between two data points on your chart.

Category:

Easy Language

Function:

TLSlopeEasy(PRICE, LEFTBAR, RIGHTBAR)

Parameters:

PRICE any bar attribute (i.e., Close, Open, etc.) or any formula

LEFTBAR refers to the bar the furthest back in time

RIGHTBAR refers to the most recent bar on your chart

Usage:

Use positive numbers to project back in time and use negative numbers to project into the future.

TLValueEasy

The TLValueEasy function returns the value of a trendline, if that trendline were to be drawn between two data points on your chart.

Category:

Easy Language

Function:

TLValueEasy(PRICE, RIGHTBAR, LEFTBAR, TRGTBAR)

Parameters:

PRICE any bar attribute (i.e., Close, Open, etc.) or any formula

RIGHTBAR refers to the most recent bar on your chart

LEFTBAR refers to the bar the furthest back in time

TRGTBAR the number of bars into the past or into the future you want to project

Usage:

Use positive numbers to project back in time and use negative numbers to project into the future.

TLAngle

This function simulates the drawing of a trend line between two points on your chart and returns the angle of that line.

Category:

Easy Language

Function:

TLAngle(PRICE1,BAR1,PRICE2,BAR2)

Parameters:

PRICE1 specifies price of trend line start

BAR1 specifies bar of trend line start

PRICE2 specifies price of trend line end

BAR2 specifies bar of trend line end

Returns:

The angle of the specified TrendLine

Usage:

To calculate the angle the function will need the bar numbers and prices for the two endpoints of the trend line. The parameters PRICE1 and PRICE2 are the starting point and ending points of the trend line. They are usually replaced by values such as Close, High, Low, etc., or are replaced with numeric series type inputs.

Note: You must offset the PRICE by its corresponding bar number. For example, if you wanted the trendline angle for a line drawn from the Close of ten bars ago to the Close of the current bar, replace the parameters PRICE1, BAR1, PRICE2, and BAR2 with the values Close[10], 10, Close[0], and 0 respectively.

The parameters BAR1 and BAR2 refer to the bar numbers of the starting and ending points of the trend line. These parameters must be replaced by positive whole numbers or be replaced with numeric simple type inputs.

The formula used is a simple rise over run calculation to obtain the slope; the formula then takes the cotangent of that slope. In fact the value returned by the TLAngle function can be obtained by taking the ArcTangent of the TLSlope function.

TL Slope

The TrendLine Slope function simulates the drawing of a trendline between two points on your chart and returns the slope of that line.

Category:

Easy Language

Function:

TLSlope(PRICE1,BAR1,PRICE2,BAR2)

Parameters:

PRICE1 specifies price of trendline start

BAR1 specifies bar of trendline start

PRICE2 specifies price of trendline end

BAR2 specifies bar of trendline end

Returns:

The slope of the specified trendline

Usage:

To calculate the angle the function will need the bar numbers and prices for the two endpoints of the trend line. The parameters PRICE2 and PRICE1 are the starting point and ending points of the trend line, respectively. They are usually replaced by values such as Close, High, Low, etc., or can be replaced with numeric series inputs.

Note: You must offset the price by its corresponding bar number. For example, if you wanted the trendline angle for a line drawn from the Close of ten bars ago to the Close of the current bar, replace the parameters PRICE1, BAR1, PRICE2, and BAR2 would be replaced with the values Close[10], 10, Close[0], and 0 respectively.

The parameters BAR2 and BAR1 refer to the bar numbers of the starting and ending points of the trendline. These parameters can be replaced by positive whole numbers or with numeric simple type inputs whos default values return positive whole numbers. The formula used is a simple rise over run calculation.

TLValue

This function simulates the drawing of a trend line between two points on your chart and then extending that trendline out into the future or back into the past.

Category:

Easy Language

Function:

TLValue(PRICE1,BAR1,PRICE2,BAR2,TGTBAR)

Parameters:

PRICE1 specifies price of trend line start

BAR1 specifies bar of trend line start

PRICE2 specifies price of trend line end

BAR2 specifies bar of trend line end

TGTBAR number of bars to extend trendline (negative number = forward in time)

Returns:

The value of the specified trend line at point TGTBAR

Usage:

The two points allow us to calculate the slope of the line and by extending that line the TLValue function is able to calculate the PRICE of the target bar in the future.

To calculate the angle the function will need the bar numbers and prices for the two endpoints of the trend line as well as the target bar out in the future or back in the past. The parameter PRICE1 and PRICE2 are the starting point and ending points of the trend line respectively. They are usually replaced by values such as Close, High, Low, etc. or replaced with numeric series type inputs.

The parameters BAR1 and BAR2 refer to the bar numbers of the starting and ending points of the trend line. These parameters must be replaced by valid numbers or replaced with numeric simple type inputs.

Note: You must offset the price by its corresponding bar number. For example, if you wanted the trend line angle for a line drawn from the Close of ten bars ago to the Close of the current bar, replace the parameters PRICE1, BAR1, PRICE2, and BAR2 would be replaced with the values Close[0], 0, Close[10], and 10 respectively.

The parameter TGTBAR must be replaced by either a positive or negative number that represents the number of bars back in the past or out in the future that the trend line is to be extended. Negative values of TGTBAR project the trendline into the future, while, positive values of TGTBAR project the trendline back into the past

The formula used to calculate the slope is a simple rise over run calculation.

Example:

The Custom One Line indicator shown in the figure below is used to return the value for a TGTBAR of ten bars in the future. The parameters PRICE1 and PRICE2 are replaced with the price Close and use 10 bars ago as your BAR2 value and zero or the current bar for

BAR1.
TrendLine Value

TrueHigh

The value the TrueHigh function returns is either the High of the current bar or the Close of the previous bar if its value is higher. The TrueHigh function is most commonly used in finding Gap Open bars. A Gap Open bar is when the Open is greater than the previous bars High.

Category:

Easy Language

Function:

TrueHigh

Returns:

The High of the current bar or the Close of the previous bar if its value is higher

Usage:

No parameters are required with this function as it only looks at two values and returns the higher of the two. The term Gap-filled is often associated with this function as this function will fill in the gap between the Close of the previous bar and the High of the current bar.

To avoid misleading gaps, the comparison made to determine the gap is best used with the TrueHigh to show a significant gap.

TrueLow

The value the TrueLow function returns is either the Low of the current bar or the Close of the previous bar if its value is lower.

Category:

Easy Language

Function:

TrueLow

Returns:

The Low of the current bar or the Close of the previous bar if its value is lower.

Usage:

No parameters are required with this function as it only looks at two values and returns the Lower of the two. The term Gap-filled is often associated with this function as this function will fill in the gap between the Close of the previous bar and the Low of the current bar.

The TrueLow function is most commonly used in finding Gap Open bars. A Gap Open bar is when the Open is lower than the previous bars Low. To avoid misleading gaps, the comparison made to determine the gap is best used with the TrueLow function to show a significant gap.

TrueRange

The True Range function returns the difference between the TrueHigh and TrueLow values. Please see the True High and True Low functions for detailed information on how their values are calculated and what they mean.

Category:

Easy Language

Function:

TrueRange

Usage:

This function is similar to Range except that it uses the TrueHigh and TrueLow values instead of the High and Low.

TrueRangeCustom

The True Range Custom function returns the difference between two variables: THIGH and TLOW.

THIGH is calculated as follows:

If the Close of one bar ago is greater than the High, then THIGH = Close of one bar ago

If not, THIGH = the High

TLOW is calculated as follows:

If the Close of one bar ago is less than the Low, then TLOW = Close of one bar ago

If not, TLOW = the Low

Category:

Easy Language

Function:

TrueRangeCustom(HPRICE, LPRICE, CPRICE)

Parameters:

HPRICE High

LPRICE Low

CPRICE Close

Usage:

This function is similar to TrueRange except that it allows greater flexibility through the use of parameters.

TypicalPrice

The Typical Price function is similar to the AvgPrice function in that it provides a more accurate value of the bar as a whole because it uses an average of the high, low and close prices of a bar rather than just using the close of the bar.

Category:

Easy Language

Function:

TypicalPrice

Usage:

The Typical Price function adds together the high, low and close of a bar and divides the total by 3.

$(H + L + C)/3$

UltimateOsc

The Ultimate Oscillator was developed by Larry Williams. Williams contends that an oscillator based on one time span is subject to a number of false signals. To combat this problem, he combines three such oscillators, each based on different time frames.

Williams states that oscillators with shorter time frames tend to peak well ahead of price, usually causing several divergences prior to the ultimate peak. On the other hand, indicators based on longer time spans do not tend to reverse direction until after market turning points.

The Ultimate Oscillator has two major interpretations. The first is the presence of divergence between the price and the oscillator itself. The second is a trend break in the oscillator itself.

Category:

Easy Language

Function:

UltimateOsc(AVG1,AVG2,AVG3)

Parameters:

AVG1	number of bars in short term average
AVG2	number of bars in intermediate term average
AVG3	number of bars in long term average

Returns:

The current value of the Ultimate Oscillator

Usage:

The parameters AVG1, AVG2, and AVG3 are replaced by the three time frames for which Williams recommends the values seven, fourteen, and twenty-eight. These are all numeric simple type parameters that can be replaced with positive whole numbers or with numeric simple type inputs.

Vega

The Vega function measures the expected change in the price on an option due to a 1 percentage point increase in volatility. Vega is always positive.

Category:

Easy Language

Function:

Vega(DAYSLEFT, STRIKE, LEVEL, RATE, VOLA, TYPE)

Parameters:

DAYSLEFT number of days left to the expiration of the option

STRIKE exercise price of the option

LEVEL price of the underlying asset

RATE currently available risk-free interest rate

VOLA annual volatility of the underlying asset

TYPE Put = 0; Call = 1

Volatility

The volatility function measures the market volatility by plotting a smoothed average of the True Range. It returns an average of the TrueRange over a specific number of bars, giving higher weight to the TrueRange of the most recent bar.

Category:

Easy Language

Function:

Volatility(LENGTH)

Parameters:

LENGTH specifies the number of trailing bars to include in the volatility calculation

Returns:

The market volatility value, as the number increases, the market is more volatile.

Usage:

Volatility is the variation in price over a specific interval (the difference between the highest and lowest prices). As the time interval being studied increases, volatility also increases to a maximum before leveling off. As prices increase, the volatility tolerance also increases. Volatility tolerance simply means that we would expect more price fluctuation as securities challenge all time highs.

We start out studying the price-volatility relationship, assuming that there is some positive correlation between the price level, time interval for measurement and potential price fluctuations. There are important advantages in finding a uniform relationship. For example, a sharp movement down that reaches or exceeds the probable limits could be used as a purchasing opportunity.

VolumeOsc

The VolumeOsc function displays the difference between a slow and fast period moving average in terms of points. The formula is:

VolumeOsc = Average(Volume, FAST) - Average(Volume, SLOW)

Category:

Easy Language

Function:

VolumeOsc(FAST,SLOW)

Parameters:

FAST time period of the slow moving average

SLOW time period of the fast moving average

Returns:

A positive or negative value

Usage:

The most common use of the indicator is to determine the likelihood of a continuation in the current move. A positive value suggests that there is sufficient participation to drive prices further in the direction of the current move. On the other hand, a negative value suggests that there is insufficient participation to drive prices further in the direction of the current move, and thus, a reversal is likely.

VolumeROC

The most common use of this function is to determine the likelihood of a continuation in the current move. The formula is:

$\text{VolumeROC} = (\text{Volume} - \text{Volume}[1]) / \text{Volume}[1]$

Category:

Easy Language

Function:

VolumeROC(LENGTH)

Parameters:

LENGTH the number of trailing bars to consider

Returns:

A positive or negative value

Usage:

A positive value suggests that there is sufficient participation to drive prices further in the direction of the current move. On the other hand, a negative value suggests that there is insufficient participation to drive prices further in the direction of the current move, and thus, a reversal is likely.

WAverage

The WAverage is a weighted moving average of the PRICE over the LENGTH specified. It is calculated by giving the more current data a heavier weight and the oldest data less weight. This makes the WAverage line follow actual prices more closely, with less of a lag than a regular Average. The WAverage is used in the same way as the average function and can give stronger and earlier indications to trend direction as it follows the most recent data closer.

Category:

Easy Language

Function:

WAverage(Price,Length)

Parameters:

PRICE specifies the data series to average

LENGTH specifies the trailing number of bars to average

Returns:

The current value of the weighted moving average

Usage:

Just as the Average function needs a PRICE and LENGTH to calculate, the WAverage function also needs the PRICE and LENGTH parameters. The PRICE is the unit you want to average, and the LENGTH is the number of units you are averaging.

WeightedClose

The Weighted Close is similar to the AvgPrice function as it calculates an average price of the bar. The WeightedClose function calculates the average price of the bar by taking the High and Low and adding it to two closes and dividing by four.

Category:

Easy Language

Function:

WeightedClose

Returns:

The value $(H+L+C+C)/4$

Usage:

This results in a average price that gives more precedence to the Close value. The Close is given the most importance of all the price information of the bar. This function is useful as a substitute price in an average function instead of using the Close.

Xaverage

The XAverage function is a weighted moving average of the prices of the last length bars. This function returns the current value of the exponentially smoothed moving average.

Category:

Easy Language

Function:

Xaverage(PRICE,LENGTH)

Parameters:

PRICE specifies which price of the asset of interest is to be used

LENGTH specifies the number of bars to consider

Returns:

The XAverage function returns a number, the weighted average of the last length bars

Usage:

When this moving average is calculated, every bars price in the data file will have an effect on the current average. The effects of the first price will never be completely removed, but its weight will continuously shrink as more and more averages are calculated.

File Writing Category

The functions in this category are available only in the PowerEditor and enable you to create and delete text files.

The function for which you want a detailed explanation:

FileAppend

Category:

File Writing

Inputs:

Name	Default	Description
Str_filename	None	Name of file to which to append text. Place name of file in parentheses. If file does not exist, file is created.
Str_text	None	String of text to append to file. Place string of text within parentheses.

Description:

The FileAppend function enables you to append a string of text to an output file. You can append several strings to a file using this function.

Usage:

The following function will send the string Values followed on the next line by the Close and Open price values of the current bar to a text file called syms.txt on your c drive.

FileAppend(C:\SYMS.TXT,Values +NewLine +

Close: +NumToStr(Close, 2) +

Open: +NumToStr(Open, 2) + NewLine);

Note: As shown in the above example, when you want to append strings spanning one or more lines, you do not need to call the FileAppend function more than once.

FileDelete

Category:

File Writing

Inputs:

Name	Default	Description
Str_filename	None	Name of file to delete. Place name of file in parentheses.

Description:

The FileDelete function enables you to delete a text file from your drive.

Usage:

FileDelete(C:\SYMS.TXT) will delete the file syms.txt from your c drive.

Fundamental (Historical) Data Category

The 12 functions within the Fundamental (Historical) Data category use three fields of fundamental data, which are supplied by your daily data vendor, along with the price data for a stock. The fields provide not just fundamental data for a specific point in time, but historically over a period of time. Specifically, the functions use the earnings-per-share (EPS), stock split (SplitValue), and dividend (Dividend) fields.

Important: These functions calculate their values based on the fundamental data provided by DialData that you have downloaded using the Omega Downloader. Therefore, these functions are for use only when you are applying them to a chart containing daily data stored in Omega Downloader format.

The 12 functions give you the ability to perform fundamental analysis in order to determine if your stock is fairly priced, undervalued or overvalued when compared to the market. The values the functions return are true for the current price series, that is, for the data you have plotted in the chart to which you are applying the function.

Each function within this category is listed below along with the correct syntax and any inputs.

Ehe function for which you want a detailed explanation:

Dividend

The Dividend function allows you to determine the amount of the stock dividend reported during a certain period. If you use the function with no input, the function returns the most recently reported dividend amount. However, if you use an input, the dividend amount for the exact period specified will be returned.

For example, entering Dividend(2) will return the reported dividend amount from 2 periods ago.

Category:

Fundamental (Historical) Data

Function:

Dividend(PeriodsAgo)

DividendCount

The DividendCount function allows you to determine the number of times, up to today, that dividends have been reported in the price series you are considering. You can also offset this function to determine the number of times dividends have been reported before today.

For instance, say today is August 30, 1996 and dividends were reported yesterday August 29, 1996 for the second time. If you apply the function with no offset today, on August 30, 1996, the function will return a value of 2. However, if you apply the function using an offset of 2 bars, DividendCount[2], then the function will return a value of 1.

Category:

Fundamental (Historical) Data

Function:

DividendCount

DividendDate

The DividendDate function returns the date on which a stock dividend was paid out. If you use the function with no input, the function returns the date of the most recently reported dividend. However, if you use an input, the date of the dividend in the exact period specified will be returned.

For example, entering DividendDate(2) will return the date on which the reported dividend from 2 periods ago was paid.

Category:

Fundamental (Historical) Data

Function:

DividendDate(PeriodsAgo)

DividendTime

The DividendTime function returns the time at which a stock dividend was paid out. If you use the function with no input, the function returns the time that the most recently reported dividend was paid out. However, if you use an input, the time that the dividend was paid out in the exact period specified will be returned.

For example, entering DividendTime(2) will return the time at which the reported dividend from 2 periods ago was paid.

Category:

Fundamental (Historical) Data

Function:

DividendTime(PeriodsAgo)

EPS

The EPS function allows you to determine what the reported earnings-per-share is during a certain period. If you use the function with no input, the function returns the most recently reported earnings-per-share. However, if you do use an input, the earnings-per-share reported in the exact period specified will be returned.

For example, entering EPS(2) will return the reported earnings-per-share from 2 periods ago.

Category:

Fundamental (Historical) Data

Function:

EPS(PeriodsAgo)

EPSCount

The EPSCount function returns the number of times that earnings-per-share has been reported up to today in the price series you are considering. You can also offset this function to determine the number of times earnings-per-share has been reported before today.

For instance, say today is August 30, 1996, and earnings-per-share were reported yesterday, August 29, 1996, for the second time. If you apply the function with no offset today, on August 30, 1996, the function will return a value of 2. However, if you apply the function using an offset of 2 bars, EPSCount[2], then the function will return a value of 1.

Category:

Fundamental (Historical) Data

Function:

EPSCount

EPSDate

The EPSDate function returns the date on which earnings-per-share was reported. If you use the function with no input, the function returns the date of the most recently reported earnings-per-share. However, if you use an input, the date that the earnings-per-share was reported for the exact period specified will be returned.

For example, entering EPSDate(2) will return the date on which the earnings-per-share from 2 periods ago was reported.

Category:

Fundamental (Historical) Data

Function:

EPSDate(PeriodsAgo)

EPSTime

The EPSTime function returns the time at which earnings-per-share was reported. If you use the function with no input, the function returns the time at which the most recently reported earnings-per-share was reported. However, if you use an input, the time at which the earnings-per-share was reported during the exact period specified will be returned.

For example, entering EPSTime(2) will return the time at which the earnings-per-share from 2 periods ago was reported.

Category:

Fundamental (Historical) Data

Function:

EPSTime(PeriodsAgo)

StockSplit

The StockSplit function allows you to determine the ratio of the stock split reported during a certain period. If you use the function with no input, the function returns the most recently reported stock split ratio. However, if you use an input, the stock split ratio for the exact period specified will be returned.

For example, entering StockSplit(2) will return the reported stock split ratio from 2 periods ago.

Category:

Fundamental (Historical) Data

Function:

StockSplit(PeriodsAgo)

StockSplitCount

The StockSplitCount function returns the number of times that stock splits have been reported up to today in the price series you are considering. You can also offset this function to determine the number of times stock splits have been reported before today.

For instance, say today is August 30, 1996, and stock splits were reported yesterday, August 29, 1996, for the second time. If you apply the function with no offset today, on August 30, 1996, the function will return a value of 2. However, if you apply the function using an offset of 2 bars, StockSplitCount[2], then the function will return a value of 1.

Category:

Fundamental (Historical) Data

Function:

StockSplitCount

StockSplitDate

The StockSplitDate function returns the date on which a stock split occurred. If you use the function with no input, the function returns the date of the most recently reported stock split. However, if you use an input, the date of the stock split in the exact period specified will be returned.

For example, entering StockSplitDate(2) will return the date on which the stock split from 2 periods ago was reported.

Category:

Fundamental (Historical) Data

Function:

StockSplitDate(PeriodsAgo)

StockSplitTime

The StockSplitTime function returns the time at which stock split was reported. If you use the function with no input, the function returns the time at which the most recently reported stock split was reported. However, if you use an input, the time at which the stock split was reported during the exact period specified will be returned.

For example, entering StockSplitTime(2) will return the time at which the stock split from 2 periods ago was reported.

Category:

Fundamental (Historical) Data

Function:

StockSplitTime(PeriodsAgo)

Fundamental (Snapshot) Data Category

The functions within the Fundamental Data (Snapshot) category use the fundamental data that is supplied by your data vendor along with the price data for a symbol. These functions give you the ability to perform fundamental analysis on the symbol in order to see how your symbol is doing in relation to the rest of the market as a whole, or what is the real or book value of your symbol. These functions provide you with the fundamental data at a specific point in time, as opposed to historically.

The functions are for use only when you are applying them to a chart based on daily data downloaded from DialData or Telescan in Omega Downloader format. If you use real-time or delayed data, but want to work with fundamental data, you should consider having an account with DialData and downloading nightly using the Omega Downloader.

These functions return the corresponding value provided by the data vendor. Each function within this category is listed below and includes a brief description of the data, the correct syntax of the function, and the corresponding field name.

Ehe function for which you want a detailed explanation:

Beta

Beta is a measure of the companys price volatility relative to the market. The Market Guide Beta is the slope of the 60-month regression line of the percentage price change of the stock relative to the percentage price change of the S&P 500, adjusted for regression tendencies reported by Blume.

The market is always the number 1. A Beta of 1 means that as the market moves, the stock should follow along. A Beta of 3 means that if the market moves up 1%, the stock should move up 3%. A Beta of .5 would mean that the stock is tired and is moving only half as much as the market.

Category:
Fundamental (Snapshot) Data
Function:
Beta
Field Name:
Beta

Beta_Down

Beta_Down is the volatility of the companys stock relative to the S&P 500 for the periods in which the S&P 500 goes down.
The returned value has the same meaning as the value returned by the Beta function, but again, it applies only to when the S&P 500 is moving down.

Category:
Fundamental (Snapshot) Data
Function:
Beta_Down
Field Name:
Beta Down

Beta_Up

Beta_Up is the volatility of the companys stock relative to the S&P 500 for the periods in which the S&P 500 goes up.
The returned value has the same meaning as the value returned by the Beta function, but again, it applies only to when the S&P 500 is moving up.

Category:
Fundamental (Snapshot) Data
Function:
Beta_Up
Field Name:
Beta Up

Book_Val_Per_Share

This function returns the Book Value Per Share, which is defined as the Common Shareholders Equity divided by the Shares Outstanding as of the end of the latest fiscal period.

Category:
Fundamental (Snapshot) Data
Function:
Book_Val_Per_Share
Field Name:
Book Value Per Share

Current_Ratio

This function returns the Current Ratio, which is defined as the Total Current Assets divided by the Total Current Liabilities. This item is not available for banks, insurance companies and other companies that do not distinguish between current- and long-term assets and liabilities.

Category:
Fundamental (Snapshot) Data
Function:
Current_Ratio
Field Name:
Current Ratio

EPS_PChng-Y-Ago

This function returns the Year Ago Earnings Per Share Percent Change value, which is the percentage change in the most recent quarterly Earnings Per Share from Continuing Operations Excluding Extraordinary items as compared to the same quarter one year ago.

Category:
Fundamental (Snapshot) Data
Function:
EPS_PChng_Y_Ago

Field Name:
EPS % Change, Year Ago

EPS_PChng-YTD

This function returns the Year-to-date Earnings Per Share Percent Change value, which is the percent change in the YTD earnings per share from Continuing Operations Excluding Extraordinary items as compared to the same YTD period of one year ago.

Category:
Fundamental (Snapshot) Data
Function:
EPS_PChng_YTD
Field Name:
EPS % Change, Year-to-date

FreeCshFlwPerShare

This function returns the Free Cash Flow Per Share value, which is the free cash flow for the trailing 12 months divided by the Average Shares Outstanding. Free cash flow is calculated from the Statement of Cash Flows as follows:

Cash From Operations - Capital Expenditures - Dividends Paid

Category:
Fundamental (Snapshot) Data
Function:
FreeCshFlwPerShare
Field Name:
Free Cash Flow Per Share

Gr_Rate_P_EPS

The Earnings per share growth rate is the compound annual growth rate of Earnings Per Share Excluding Extraordinary Items and Discontinued Operations. If the most recent fiscal year earnings per share value is negative, the result is not meaningful. It is calculated for 3 years whenever 4 years of earnings are available and positive. If the required 4 years are not available for a given company, the calculation is performed over a shorter time period dependent upon the number of years of earnings available. To determine how many years were used in the calculation see the EPS Growth Rate, Number of Years (G_Rate_EPS_NY).

Category:
Fundamental (Snapshot) Data
Function:
Gr_Rate_P_EPS
Field Name:
Growth Rate % EPS

G_Rate_EPS_NY

This is the number of years over which the Earnings Per Share Growth Rate is calculated.

Category:
Fundamental (Snapshot) Data
Function:
G_Rate_EPS_NY
Field Name:
Growth Rate Earnings Per Share Number of Years

G_Rate_NT_IN_NY

This is & number of years over which the Net Income Growth Rate is calculated.

Category:
Fundamental (Snapshot) Data

Function:
G_Rate_NT_IN_NY
Field Name:
Growth Rate Net Income Number of Years

G_Rate_P_Net_Inc

The Earnings per share growth rate is the compound annual growth rate of Earnings Per Share Excluding Extraordinary Items and Discontinued Operations. If the most recent fiscal year earnings per share value is negative, the result is not meaningful. It is calculated for 3 years whenever 4 years of earnings are available and positive. If the required 4 years are not available for any given company, the calculation is performed over a shorter time period dependent upon the number of years of earnings available. To determine how many years were @ in the calculation see Net IncomeGrowth Rate, Number of Years (G_Rate_NT_IN_NY).

Category:

Fundamental (Snapshot) Data

Function:

G_Rate_P_Net_Inc

Field Name:

Growth Rate % Net Income

Inst_Percent_Held

The percent of common stock held by all the reporting institutions as a group:

$\text{Total Shares held by Institutions} / \text{Total Shares Outstanding} * 100$

Category:

Fundamental (Snapshot) Data

Function:

Inst_Percent_Held

Field Name:

Institutional % Held

Net_Profit_Margin

The Net Profit Margin is the Income after Taxes divided by the Total Revenue and is expressed as a percentage, and calculated over the Trailing Twelve Months. Most banks and finance companies do not report revenues when they announce their preliminary quarterly earnings in the press. When this happens the quarterly and trailing twelve month values will not be available.

Category:

Fundamental (Snapshot) Data

Function:

Net_Profit_Margin

Field Name:

Net Profit Margin

Quick_Ratio

The Quick Ratio, also known as the Acid Test Ratio, is defined as:

$\{\text{Cash} + \text{Short Term Investments} + \text{Accounts Receivable}\} / \text{Total Current Liabilities}$

Category:

Fundamental (Snapshot) Data

Function:

Quick_Ratio

Field Name:

Quick Ratio

Ret_On_Avg_Equity

The Return on Common Equity is the income available to common stockholders divided by the Average Common Equity and is expressed as a percentage.

Category:

Fundamental (Snapshot) Data

Function:

Ret_On_Avg_Equity

Field Name:

Return on Average Equity

TtlDbt_By_NetAssts

This ratio is the Total Debt (long- and short-term) divided by Total Assets.

Category:
Fundamental (Snapshot) Data
Function:
TtlDbt_By_NetAssts
Field Name:
Total Debt / Net Assets

Last_Split_Date

The Last Split Date is the date on which the stock last split.

Category:
Fundamental (Snapshot) Data
Function:
Last_Split_Date
Field Name:
Last Split Date

Last_Split_Fact

The Last Split Factor denotes the size of the last stock split. For example, a split factor of 2.0000 indicates a 2-for-1 stock split or a 100% stock dividend.

Category:
Fundamental (Snapshot) Data
Function:
Last_Split_Fact
Field Name:
Last Split Factor

Dividend_Yield

This function returns the Indicated Annual Dividend Rate, which is the total of the expected dividend payments over the next twelve months. It is generally the most recent cash dividend paid or declared multiplied by the dividend payment frequency, plus any regular dividends.

Category:
Fundamental (Snapshot) Data
Function:
Dividend_Yield
Field Name:
Dividend Share

Price_To_Book

This function returns the Primary EPS Excluding Extraordinary Items and Discontinued Operations, which is the adjusted income available to common stockholders divided by the primary average shares outstanding.

Category:
Fundamental (Snapshot) Data
Function:
Price_To_Book
Field Name:
EPS (Excluding Extraordinary Items)

SGA_Exp_By_NetSales

The total revenue divided by the number of outstanding shares.

Category:
Fundamental (Snapshot) Data
Function:
SGA_Exp_By_NetSales
Field Name:
Revenue Per Share

Math & Trig Category

The functions in the Math & Trig category each perform common yet valuable mathematical and trigonometric calculations such as the rounding of numbers, square roots, and the natural log of a

number. Each mathematical and trigonometric function is listed next. The listing includes a description of the function and the appropriate syntax.

The function for which you want a detailed explanation:

AbsValue

The AbsValue function returns the absolute value of any numeric expression; the absolute value of a number is its positive equivalent.

Category:

Math & Trig

Function:

AbsValue(num)

Examples:

AbsValue(-45) returns +45

AbsValue(26) returns +26

AbsValue(Value1) returns the positive equivalent of Value1

AbsValue(C-C[1]) returns the net change as a positive number

ArcTangent

Where x is the tangent value of an angle, ArcTangent returns the angle in degrees for x. The tangent is equivalent to sine/cosine.

Category:

Math & Trig

Function:

ArcTangent(x)

AvgList

Returns the average of all items in the list. The maximum number of items allowed in a list is 25. In list functions, the number of parameters, up to a maximum of 25, is determined by the user. An item may be any numeric expression.

Category:

Math & Trig

Function:

AvgList(num,num,num, etc...)

Examples:

AvgList(5,7,10) returns 7.333...

AvgList(Value1,Close of today,Truehigh[1]) returns the average of the three items.

Ceiling

Returns the lowest integer greater than a number (num). Also see the Floor function.

Category:

Math & Trig

Function:

Ceiling(Num)

Examples:

Ceiling(6.8) returns 7

Ceiling(-12.4) returns -12

Cosine

Returns the cosine of x.

Category:

Math & Trig

Function:

Cosinenum)

Example:

Cosine(Value1) assuming Value1 = 8 degrees, returns .990268, the cosine of Value1

CoTangent

Returns the co-tangent of x.

Category:

Math & Trig

Function:

CoTangent(num)

Example:

CoTangent(Value1) assuming Value1 = 8 degrees, returns 82.875, the co-tangent of Value1.

ExpValue

Returns the exponential value of x, the value e raised to the power of x, where e is the base of the natural logarithm.

Category:

Math & Trig

Function:

ExpValue(num)

Example:

ExpValue(Value1) assuming Value1 = 8, returns 2980.957987, the exponential of Value1.

Floor

Returns the highest integer less than Num. Also see the Ceiling function.

Category:

Math & Trig

Function:

Floor(Num)

Example:

Floor(6.8) returns 6

Floor(-12.4) returns -13

FracPortion

Returns the fractional, or decimal portion of a number (num). Also see the functions: IntPortion and Mod.

Category:

Math & Trig

Function:

FracPortion(num)

Example:

FracPortion(Value1) assuming Value1 = 4.56, returns .56, the fractional part of 4.56

IntPortion

Returns the integer portion of a number (num). Num may be any numeric expression. Also see the functions: FracPortion and Mod.

Category:

Math & Trig

Function:

IntPortion(num)

Example:

IntPortion(4.5) returns -4 as the integer portion of -4.5

Log

Returns the natural logarithm of a number (num). Num may be any numeric expression.

Category:

Math & Trig

Function:

Log(num)

Example:

Log(4) returns 1.38629436

MaxList

Returns the maximum value of a list of items. The maximum number of items allowed in a list is 25. In list functions, the number of parameters, up to a maximum of 25, is determined by the user. An item may be any numeric expression.

Category:

Math & Trig

Function:

MaxList(num,num,num, etc...)

Example:

MaxList(RSI(Close,14),RSI(Close,14)[5],RSI(Close,14)[10]) returns the maximum RSI in the list

MaxList(Highest(Close,10),Highest(Open,10)) returns the max. Highest in the list

MaxList2

Returns the 2nd maximum value of a list of items. The maximum number of items allowed in a list is 25. In list functions, the number of parameters, up to a maximum of 25, is determined by the user. An item may be any numeric expression.

Category:

Math & Trig

Function:

MaxList2(num,num,num, etc...)

Example:

MaxList2(RSI(Close,14),RSI(Close,14)[5],RSI(Close,14)[10]) returns the 2nd maximum RSI in the list

MaxList2(Highest(Close,10),Highest(Open,10)) returns the 2nd maximum Highest in the list

MinList

Returns the minimum value of a list of items. The maximum number of items allowed in a list is 25. In list functions, the number of parameters, up to a maximum of 25, is determined by the user. An item may be any numeric expression.

Category:

Math & Trig

Function:

MinList(num,num,num,etc...)

Example:

MinList(RSI(Close,14),RSI(Close,14)[5],RSI(Close,14)[10]) returns the minimum RSI in the list

MinList(Highest(Close,10),Highest(Open,10)) returns the min. Highest in the list

MinList2

Returns the 2nd minimum value of a list of items. The maximum number of items allowed in a list is 25. In list functions, the number of parameters, up to a maximum of 25, is determined by the user. An item may be any numeric expression.

Category:

Math & Trig

Function:

MinList2(num,num,num,etc...)

Example:

MinList2(RSI(Close,14),RSI(Close,14)[5],RSI(Close,14)[10]) returns the 2nd minimum RSI in the list

MinList2(Highest(Close,10),Highest(Open,10)) returns the 2nd minimum Highest in the list

Mod

This function performs modulus arithmetic. It calculates the remainder (fractional portion) of a number (num) divided by the specified base.

Category:

Math & Trig

Function:

Mod(num,base)

Example:

Mod (12,10) returns 2

Mod (-4,3) returns -1

Neg

Returns the negative value of the absolute value of num.

Category:

Math & Trig

Function:

Neg(num)

Example:

Neg (12) returns -12
Neg (-4) returns -4

NthMaxList

Returns the Nth highest value in list.
Category:
Math & Trig
Function:
NthMaxList(N,num,num,num,...)
Example:
NthMaxList(2,8,10,1) returns 8

NthMinList

Returns the Nth lowest value in list.
Category:
Math & Trig
Function:
NthMinList(N,num,num,num...)

Pos

Returns positive value of number. See also AbsValue and Neg.
Category:
Math & Trig
Function:
Pos(num)
Example:
Pos (-7) returns +7
Pos (10) returns +10

Power

Computes a number (num) raised to the specified base power.
Category:
Math & Trig
Function:
Power(num, base)
Example:
Power (10,3) returns 1000

Round

Returns num rounded to the nearest whole number.
Category:
Math & Trig
Function:
Round(num,precision)
Example:
Round(4.56,1) returns 4.6
Round(4.40,0) returns 4
Round(0.56,0) returns 1
Round(-4.56,0) returns -5

Sign

Returns a +1 when a positive number is identified, a -1 when a negative number is identified, and a 0 when a zero is identified.
Category:
Math & Trig
Function:
Sign(num)
Example:
Sign(Value1) assuming that Value1 is -.89, returns a -1

Sine

Returns the sine of a number (num).

Category:

Math & Trig

Function:

Sine(num)

Example:

Sine(Value1) assuming Value1 is 8 degrees, returns 0.139173100, the sine of Value1

Square

Returns the square of a number (num). Num may be any numeric expression.

Category:

Math & Trig

Function:

Square(num)

Example:

Square(4) returns 16

SquareRoot

Returns the square root of a number (num). Num may be any numeric expression.

Category:

Math & Trig

Function:

SquareRoot(num)

Example:

SquareRoot(16) returns 4

SumList

Returns the cumulative total of the items in the list. An item in the list may be any numeric expression. The maximum number of items allowed in a list is 25. In list functions, the number of parameters, up to a maximum of 25, is determined by the user.

Category:

Math & Trig

Function:

SumList(num,num,num, etc...)

Example:

SumList(Value1,Value2) If Value1 = 10 and Value2 = 5, returns 15

Tangent

Returns the tangent of x.

Category:

Math & Trig

Function:

Tangent(num)

Example:

Tangent(Value1) assuming Value1 = 8 degrees, returns .1405408, the tangent of Value1

Multimedia Category

The functions in the Multimedia category enable you to work with movie files (files in .avi format). You can assign up to 15 movie files to a movie chain. The movie files in the movie chain will be played back seamlessly.

The function for which you want a detailed explanation:

MakeNewMovieRef

Category:

Multimedia

Description:

The MakeNewMovieRef function returns a new reference to a movie chain. Once you have a movie reference, you can create a movie chain by adding up to 15 movie files to the movie reference. You create the movie chain by using the AddToMovieChain function.

Usage:

The function will return an alphanumeric movie reference, for example, mvRef8.

AddToMovieChain

Category:

Multimedia

Inputs:

Name	Default	Description
MovieRef	None	Movie reference returned by MakeNewMovieRef function.
StrPath	None	Full path and name of movie file to add to chain.

Description:

The AddToMovieChain function enables you to add movies to the movie reference, thereby creating a movie chain. You can add up to 15 movies to a movie chain. You use the function once for every movie file you want to add to the chain.

Usage:

AddToMovieChain(mvRef8,c:\bullish.avi) will add the movie file bullish.avi to the movie reference mvRef8.

GetCDROMDrive

The GetCDROMDrive function will return a string containing the drive letter of the CD ROM drive that is connected to your computer. If you have multiple CD ROM drives, it will return the first one it finds.

Category:

Multimedia

Function:

GetCDROMDrive

Returns:

Letter of first CD ROM drive found

PlayMovieChain

Category:

Multimedia

Inputs:

Name	Default	Description
MovieRef	None	Movie reference returned by MakeNewMovieRef function.

Description:

The PlayMovieChain function enables you to play the movie chain you have created using the movie reference and movie files.

Usage:

PlayMovieChain(mvRef8) will play the movie chain referenced by mvRef8.

PlaySound

Category:

Multimedia

Inputs:

Name	Default	Description
StrPath	None	Full path and name of sound file to play.

Description:

The PlaySound function enables you to play a sound file from your hard drive when an event occurs.

Usage:

PlaySound(c:\windows\hiscore.wav) will play the sound contained in the hiscore file.

Pager Category

The two functions in the Pager category allow you to send an alphanumeric string to a pager. These functions are relevant only when you have enabled TradeStations optional pager notification feature.

These functions can only be accessed by means of the PowerEditor; they are not available when using the QuickEditor.

The function for which you want a detailed explanation:

Pager_DefaultName

Category:

Pager

Function:

Pager_DefaultName

Usage:

Returns the Subscriber Name entered by the TradeStation user in the TradeStation Pager dialog.

Example:

If the TradeStation Pager dialog contains Fred K in the Subscriber Name edit box, Pager_DefaultName will return Fred K.

Pager_Send

Category:

Pager

Function:

Pager_Send(str_name,str_msg)

Usage:

Sends a message to a subscriber by means of a call to an alphanumeric pager, if the pager notification feature in TradeStation is enabled.

Example:

Pager_Send(Fred K, Received first down-tick) would send the message Received first down-tick to the subscriber defined as Fred K.

Performance Information Category

Performance Information functions provide information on not just a specific position but about the overall performance of a system. These functions, like Position Information functions, are not practical for use in any analysis technique other than trading systems. They return the information that is normally viewed in your charting applications System Report. They refer to trades only, not to positions.

A trade is a position that is completely liquidated (closed out). All numbers are based on closed out trades only and are updated after the end of the closed out trade.

The listing of each Performance Information function includes a brief description of the purpose of the function, the correct syntax, and the corresponding formula.

The function for which you want a detailed explanation:

AvgBarsLosTrade

Returns a number representing the average number of bars in closed out losing trades.

Category:

Performance Information

Function:

@AvgBarsLosTrade

Formula:

@TotalBarsLosTrades / @NumLosTrades

AvgBarsWinTrade

Returns a number representing the average number of bars in closed out winning trades.

Category:

Performance Information

Function:

@AvgBarsWinTrade

Formula:

@TotalBarsWinTrades / NumLosTrades

GrossLoss

Returns the cumulative dollar total of all closed out losing trades.

Category:

Performance Information

Function:

@GrossLoss

GrossProfit

Returns the cumulative dollar total of all closed out winning trades.

Category:

Performance Information

Function:

@GrossProfit

LargestLosTrade

Returns the dollar amount of the largest closed out losing trade.

Category:

Performance Information

Function:

@LargestLosTrade

LargestWinTrade

Returns the dollar amount of the largest closed out winning trade.

Category:

Performance Information

Function:

LargestLosTrade

MaxConsecLosers

Returns a number representing the longest string of consecutive closed out losing trades.

Category:

Performance Information

Function:

MaxConsecLosers

MaxConsecWinners

Returns a number representing the longest string of consecutive closed out winning trades.

Category:

Performance Information

Function:

@MaxConsecWinners

MaxContractsHeld

Returns the maximum number of contracts held at any one time.

Category:

Performance Information

Function:

MaxContractsHeld

MaxIDDrawdown

Returns the largest equity dip sustained based on closed out trades plus open position losses on the Close of each bar. Maximum intra-day drawdown thus returns the true number of dollars needed to sustain the largest equity dip. This function is updated bar by bar.

Category:

Performance Information

Function:

MaxIDDrawdown

NetProfit

Cumulative dollar total on all closed out trades both winning and losing. Returns the difference between gross profit and gross loss.

Category:

Performance Information

Function:

NetProfit

Formula:

GrossProfit - @GrossLoss

NumLosTrades

Returns a total count of the number of closed out losing trades.

Category:

Performance Information

Function:

NumLosTrades

NumWinTrades

Returns a total count of the number of closed out winning trades.

Category:

Performance Information

Function:

NumWinTrades

PercentProfit

Returns the percentage of all closed out trades which were winners.

Category:

Performance Information

Function:

PercentProfit

Formula:

$\text{NumWinTrades} / \text{@TotalTrades}$

TotalBarsLosTrades

Returns the cumulative total number of bars in closed out losing trades.

Category:

Performance Information

Function:

TotalBarsLosTrades

TotalBarsWinTrades

Returns the cumulative total number of bars in closed out winning trades.

Category:

Performance Information

Function:

TotalBarsWinTrades

TotalTrades

Returns the cumulative total number of all closed out trades.

Category:

Performance Information

Function:

TotalTrades

Formula:

$\text{NumLosTrades} + \text{@NumWinTrades}$

Position Information Category

A position refers to trades all of which are in the same direction. For example, if you have three long entries without any exits, you are still in the same position. This is an example of scaling into a position. When you exit, it always exits all entries. Your position changes when you move from:

Long to Short

Long to Flat

Short to Flat

Short to Long

Flat to Long

Flat to Short

Keep the following in mind when using the Position Information functions:

One position may consist of any number of individual entries.

If there is no position, all functions return 0. If you pass a 5, for example, but have only had three positions, all of these functions will return 0.

These functions cannot use Data2 - Data50.

All functions are based on and are current as of the Close of the previous bar.

In the Position Information functions, entry refers to the first entry in a position and exit refers to the exit that closed out your position. The position number ago (PosNumAgo) logic is the same.

The only exceptions are @AvgEntryPrice, @CurrentContracts, and @CurrentEntries, which do not have an input because they always refer to the current Open position. In these functions, PosNumAgo must be a number between 0 and 10, and is equivalent to the position number ago (0 = current Open position; 10 = 10 previous positions ago).

In @ExitDate, @ExitPrice and @ExitTime, PosNumAgo must be a number between 1 and 10, and PosNumAgo is optional; the default is 0.

The function for which you want a detailed explanation:

AvgEntryPrice

Returns the average entry price for all open entries in the current position. The function is valid only for the current position; therefore, you cannot pass PosNumAgo as an input.

Category:

Position Information

Function:

AvgEntryPrice

Example:

For example, if you had bought 1 contract on your first entry at 250.00 and 3 contracts on your second entry at 260.00, @AvgEntryPrice would be 257.50.

BarsSinceEntry

Returns the number of bars which have gone by since you originally entered the position.

PosNumAgo must be a number between 0 and 10 representing the number for positions ago.

The value is zero if you have not had PosNumAgo number of positions.

The following returns 0 if you have only had 5 positions: @BarsSinceEntry(6)

Category:

Position Information

Function:

@BarsSinceEntry(PosNumAgo)

Example:

Assuming you have an open position, the following returns the number of bars ago since you first entered the current position: @BarsSinceEntry(0)

BarsSinceExit

Returns the number of bars which have gone by since you completely closed out the position. PosNumAgo must be a number between 1 and 10 representing the number of positions ago. The value returned is zero if you have not had PosNumAgo number of positions.

The following returns 0 if you have only had 5 positions:@BarsSinceExit(6)

Category:

Position Information

Function:

BarsSinceExit(PosNumAgo)

Example:

The following returns the number of bars which have elapsed since you completely closed out 1 position ago: @BarsSinceExit(1)

CurrentContracts

Returns the number of contracts open within the current position. This function is valid for the current position only. You cannot pass PosNumAgo as an input.

Category:

Position Information

Function:

CurrentContracts

CurrentEntries

Returns the number of entries which are still open within the current position. This function is valid only for the current position. You cannot pass PosNumAgo as an input.

Category:

Position Information

Function:
CurrentEntries
Example:

CurrentEntries refers to the number of entries, not the number of contracts. If you bought 5 contracts on your first entry and 10 contracts on your second entry, you would have made 2 entries.

EntryDate

Returns entry date of PosNumAgo, the original entry into the position. This function does not refer to subsequent entries which may have scaled you into a position. If no parameter is passed or 0, this function will return 0 if current position is flat. In addition, if 1 is passed, for example, this function will return 0 if there was no previous position.

Category:
Position Information
Function:
EntryDate(PosNumAgo)

Example:
Assuming you have a current open position in the market which was entered on July 12, 1991, the function returns 910712: @EntryDate

EntryPrice

This function returns the original entry price of PosNumAgo position ago in decimal format. This function does not refer to subsequent entries which may have scaled you into a position. The value returned is zero if you have not had PosNumAgo number of positions. This function should never be used in an entry signal with an input value of zero. This is because the function is only updated after you enter the market.

When writing your own exit signals, you can use @EntryPrice to refer to the entry price of a certain position. Returns the EntryPrice of PosNumAgo original entry.

Category:
Position Information
Function:
EntryPrice(PosNumAgo)

Example:
Assuming you are long in bonds from 102 16/32, the function returns a value of 102.50:
@EntryPrice

EntryTime

Returns the time of the entry par PosNumAgo original position. This function does not refer to subsequent entries which may have scaled you into a position. This function is used only with intra-day data.

Category:
Position Information
Function:
EntryTime(PosNumAgo)

ExitDate

Returns the date in YYMMDD format when PosNumAgo position was completely closed out. This function refers only to the final exit that completely closed out a position, not to other exits which may have scaled you out of the position. Note: PosNumAgo must be ≥ 1 ; the valid parameters are 3-10.

Category:
Position Information
Function:
@ExitDate(PosNumAgo)

ExitPrice

Returns the exit price when PosNumAgo position was completely closed out. This function refers only to the final exit that completely closed out a position, not to other exits which may have scaled you out of the position. Note: PosNumAgo must be ≥ 1 ; the valid parameters are 3-10.

Function:

@ExitPrice(PosNumAgo)

Example:

If the last exit of a position occurred at 245.80, the following function returns 245.80:

@ExitPrice(1)

ExitTime

Returns the time of the bar PosNumAgo when you completely closed out the position. This function refers only to the final exit that completely closed out a position, not to other exits which may have scaled you out of the position. This is used only with intra-day data. Note: PosNumAgo must be ≥ 1 ; the valid parameters are 3-10.

Category:

Position Information

Function:

@ExitTime(PosNumAgo)

MarketPosition

Returns whether position was: 0 = flat; 1 = long; -1 = short

To avoid two long positions back to back, place the following as a condition in your long entry signal:

Condition1 = MarketPosition(0) = -1 or

(MarketPosition(0) = 0 and MarketPosition(1) $< >$ 1);

To avoid two short positions back to back, place the following as a condition in your short entry signal:

Condition 1 = MarketPosition(0) = 1 or (MarketPosition(0) = 0 and MarketPosition(1) $< >$ -1);

Assuming the market position was short, the function will return a value of -1.

Note: This function does not return the number of contracts long or short. For that information, use the @CurrentContracts (for current position) or @MaxContracts function.

Category:

Position Information

Function:

MarketPosition(PosNumAgo)

MaxContracts

Returns the maximum number of contracts held at any one time during the life of that position.

Category:

Position Information

Function:

MaxContracts(PosNumAgo)

Example:

If the maximum number of contracts held open at the same time 2 positions ago was 20, the following returns 20:@MaxContracts(2)

MaxEntries

Returns the maximum number of open entries at any one time in that position.

Category:

Position Information

Function:

@MaxEntries(PosNumAgo)

MaxPositionLoss

Returns the maximum loss during the position.

Category:

Position Information

Function:

MaxPositionLoss(PosNumAgo)

MaxPositionProfit

Returns the maximum profit during the position.

Category:

Position Information
Function:
MaxPositionProfit(PosNumAgo)

OpenPositionProfit

Returns the profit for the open position.
Category:
Position Information
Function:
OpenPositionProfit

PositionProfit

Returns the realized profit or loss for PosNumAgo position. If input value = 0, this function returns the open position profit or loss as of the current bar. If input value >= 1, then this function returns the closed-out position profit or loss.
This function will return a negative number if the position was a loss and a positive number if the position was profitable. PositionProfit(0) is equivalent to the Open Position Profit/Loss shown on the Performance Summary.
Category:
Position Information
Function:
PositionProfit(PosNumAgo)

String Related Category

The functions in the String Related category are functions that enable you to work with strings (both numeric strings and text strings). These functions can only be accessed by means of the PowerEditor; they are not available when using the QuickEditor.
The function for which you want a detailed explanation:

GetSymbolName

Category:
String Related
Function:
GetSymbolName
Usage:
Returns the name of the symbol within a chart.
Example:
GetSymbolName applied to a chart with Microsoft price data plotted will return MSFT.

InStr

Category:
String Related
Function:
InStr(string1,string2)
Usage:
Returns the location of the first found occurrence of string2 within string1
Example:
Instr(HAL,X) returns 0 because string2 was not found in string1, whereas Instr(Off) returns 2, because the first occurrence of the letter f within Off is the second position.

LeftStr

Category:
String Related
Function:
LeftStr(string,size)
Usage:
Returns the leftmost portion of the string, up to the number of characters specified by SIZE.
Example:
LeftStr(Hi There,2) returns Hi.

LowerStr

Category:
String Related
Function:
LowerStr(str)
Usage:
Returns the lowercase version of the string passed.
Example:
LowerStr(MSFT) returns msft.

MidStr

Category:
String Related
Function:
MidStr(string,pos,size)
Usage:
Returns a specified portion of the string.
Example:
MidStr(A bearish market,3,7) returns the seven characters beginning with the third character, or in this case: bearish.

NewLine

Category:
String Related
Function:
NewLine
Usage:
Used to embed a carriage return and linefeed command in output strings.
Example:
Placing NewLine in an analysis technique would return a carriage return and linefeed command.

NumToStr

Category:
String Related
Function:
NumToStr(num,decimals)
Usage:
Returns the string, converted from NUM, using specified decimal places.
Example:
NumToStr(10.23,30) returns the string 10.230.

RightStr

Category:
String Related
Function:
RightStr(string,size)
Usage:
Returns the rightmost portion of the string, as specified by SIZE.
Example:
RightStr(Hello, World!,6) returns the string World!.

Spaces

Category:
String Related
Function:
Spaces(count)
Usage:
Returns a string of empty spaces, which can be used for padding output.
Example:
Spaces(5) returns , in this case, 5 empty spaces.

StrLen

Category:

String Related

Function:

StrLen(str)

Usage:

Returns the number of characters in a string.

Example:

StrLen(1357908642) returns 10 because there are ten characters in the string, and StrLen() returns 0, because there are no characters in the string.

StrToNum

Category:

String Related

Function:

StrToNum(string)

Usage:

Returns the numerical value of the string.

Example:

StrToNum(23.10) returns 23.1 and StrNum(Hi) returns 0 because Hi has no numerical string.

UpperStr

Category:

String Related

Function:

UpperStr(str)

Usage:

Returns the uppercase version of the string passed.

Example:

UpperStr(msft) returns MSFT.

System Information (for plots) Category

Previously, you could not plot equity information generated by a trading system. You could not graphically represent how much money you were losing or making based on your system. However, beginning with TradeStation 4.0, Omega Research has made available to users a category of functions that can be called from an indicator and used to visibly display system equity information. These functions are listed below, along with a brief description of their purpose. All the functions apply to equity instruments only.

The function for which you want a detailed explanation:

AvgEntryPrice

Illustrates the average entry price for all open equity entries in the current position; the function is valid only for the current position.

Category:

System Information (for plots)

Function:

I_AvgEntryPrice

ClosedEquity

This function illustrates the equity, the profit or loss, realized when your position is exited. To display your profit or loss along your entire position, you should use the I_OpenEquity function.

Category:

System Information (for plots)

Function:

I_ClosedEquity

CurrentContracts

The I_CurrentContracts function illustrates the number of shares held in the current open position.

Category:
System Information (for plots)
Function:
I_CurrentContracts

MarketPosition

This function returns your current market position: long, short or flat. The values returned are as follows:

Short = -1
Long = 1
Flat = 0
Function:
I_MarketPosition

OpenEquity

This function illustrates your profit or loss throughout the position, not just upon the closing as with the I_ClosedEquity function.

Category:
System Information (for plots)
Function:
I_OpenEquity

System Information Category

The functions within the System Information category all allow you to read and gather the particular settings defined within your charting application. The functions do not perform any calculations, rather, their purpose is strictly to gather information on your charting applications settings such as commission, margins on futures, slippage, etc. If the user has not changed the setting, the value returned by the function will be the charting applications default value.

The function for which you want a detailed explanation:

BreakEvenStopFloor

This function returns the setting defined for the Breakeven Stop floor level. The Breakeven Stop is activated when the maximum open position profit exceeds your selected per contract or per position floor level. Once the floor level is reached, a stop is placed at your entry price for the position (average entry price if multiple entries). If your floor is never penetrated, the stop will not become active, therefore, to limit loss, you might want to use the Money Management Stop along with the Breakeven Stop.

Category:
System Information
Function:
BreakEvenStopFloor

Commission

The Commission function returns the round-turn commission value in dollars specified in the system on a per-contract basis. Stock traders should use zero (0) commission.

Category:
System Information
Function:
Commission

Margin

The Margin function returns the margin value of futures contracts specified in the system.

Category:
System Information
Function:
Margin

MoneyMgtStopAmt

This function returns the setting defined in your charting application as the Money Management Stop amount.

The Money Management Stop lets you set the maximum amount of money you are willing to risk per contract on any one position. For example, if you enter \$500 you will automatically close out an entire position if a per contract or per position loss of \$500 or more is sustained.

Since your charting application knows the dollar value of each one-point move for the security to which the system is applied, it is able to give you the exact point at which to place your protective stop once a position has been entered.

Category:

System Information

Function:

MoneyMgtStopAmt

ProfitTargetStop

This function returns the Profit Target setting, which is specified in your charting application in dollars per contract or per position. The Profit Target is the dollar amount at which you will automatically close out a position and take profits.

If the specified profit level is never reached, the stop will not take effect. Therefore, to limit loss, users will sometimes use the Profit Target together with the Money Management Stop.

Category:

System Information

Function:

ProfitTargetStop

Slippage

The Slippage function returns the slippage value specified in the system. Slippage is the difference in dollars, on a per contract basis between the order price and the price at which the order was filled. Slippage is deducted from profitable trades and added to losing trades. This slippage amount is used to compensate for any real-time slippage in a trade.

Category:

System Information

Function:

Slippage

TrailingStopAmt

This function returns the dollar amount specified within your charting application as the Dollar Risk Trailing Stop Amount.

The Dollar Risk Trailing Stop amount allows you to indicate how much of the maximum open position profit you are willing to give back before the position is automatically closed out.

The maximum profit is calculated from the point of entry using the highest High (if long) or the lowest Low (if short). The dollar amount of profit per contract or per position you are willing to risk is then subtracted and the trailing stop is placed at that point.

For example, assume that a Dollar Risk Trailing Stop is placed for the amount of \$500. A protective stop would be placed as soon as the open position profit retraced \$500 from its highest point since the position was entered.

Category:

System Information

Function:

TrailingStopAmt

TrailingStopFloor

This function returns the percentage specified in your charting application as the Trailing Stop profit level, or floor level.

The Percent Risk Trailing Stop allows you to indicate what percent of the maximum position profit you are willing to give back before the position is automatically closed out. The maximum profit is calculated from the point of entry using the highest High (if long) or lowest Low (if short). The percent of this amount you are willing to risk is then subtracted and the trailing stop is placed at that point.

The Percent Risk Trailing Stop requires a profit level (floor level). This level must be reached before the stop can be triggered. For example, assume that a Percent Trailing Stop is placed at 20% with a floor stop of \$500. Once profits exceed the floor value of \$500, the stop will become active. The stop is then placed for the maximum open position profit since entry minus 20%.

If the maximum open position profit for the trade does not exceed the floor level, this stop does not take effect. Consequently, the Percent Trailing Stop only locks in profit; it does not exit a position if your floor level was never reached. In order to lock in a profit and limit loss, use the Percent Trailing Stop together with the Money Management Stop.

Category:

System Information

Function:

TrailingStopFloor

TrailingStopPct

This function returns the percentage specified in your charting application as the Percent Risk Trailing Stop.

The Percent Risk Trailing Stop allows you to indicate what percent of the maximum position profit you are willing to give back before the position is automatically closed out. The maximum profit is calculated from the point of entry using the highest High (if long) or lowest Low (if short). The percent of this amount you are willing to risk is then subtracted and the trailing stop is placed at that point.

The Percent Risk Trailing Stop requires a profit level (floor level). This level must be reached before the stop can be triggered. For example, assume that a Percent Trailing Stop is placed at 20% with a floor stop of \$500. Once profits exceed the floor value of \$500, the stop will become active. The stop is then placed for the maximum open position profit since entry minus 20%.

If the maximum open position profit for the trade does not exceed the floor level, this stop does not take effect. Consequently, the Percent Trailing Stop only locks in profit; it does not exit a position if your floor level was never reached. In order to lock in a profit and limit loss, use the Percent Trailing Stop together with the Money Management Stop.

Category:

System Information

Function:

TrailingStopPct

Systems to Include Category

The System to Include category technically does not contain functions. Rather, it is made up of actual trading systems found in your TradeStation trading system library. The fact that you can access them and use them as functions allows you to place them within another system.

These systems can only be accessed by means of the PowerEditor; they are not available when using the QuickEditor.

The function for which you want a detailed explanation:

System: CCIAvgCrossover

Category:

System to Include

Inputs:

Name	Default	Description
CCILen	20	Number of bars used in calculation
Avg1Len	10	Number of bars used in calculation
Avg2Len	20	Number of bars used in calculation

Description:

The Commodity Channel Index (CCI) system is best used with contracts that display cyclical or seasonal characteristics. It is formulated to detect the beginning and ending of these cycles by incorporating a moving average together with a divisor, which reflects both possible and actual trading ranges. The final index measures the deviation from normal which indicates major changes in market trend.

Usage:

The CCI is designed so that 70% - 80% of all price fluctuations fall between +100 and -100. When the CCI exceeds 100, you establish a long position; when it falls below -100, you go short. At what point above +100 you go long, or below -100 you go short, is your decision based on particular market analysis. For example, you may decide that -125 suggests going short, while a rise to -75 suggests liquidating a short position.

Note: Many traders use this indicator in the exact opposite way. They use it as an overbought/oversold indicator with 100 or greater indicating that the market is overbought, and -100 or less that the market is oversold.

System: ChannelBrkIntraBar

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

Length10		Number of bars for price channel
----------	--	----------------------------------

Description:

Channel Breakouts are one of the most well-known and popular trading techniques. They are based on the logic that when the market breaks the upper/lower channel of previous prices, a new trend has begun.

Usage:

In common usage, a breakout of the upper channel signifies the beginning of a strong upward move while a breakout of the lower channel signifies the beginning of a strong downward move.

System: ChannelBrkOnClose

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

Length10		Number of bars for price channel
----------	--	----------------------------------

Description:

Channel Breakouts are one of the most well-known and popular trading techniques. They are based on the logic that when the market breaks the upper/lower channel of previous prices, a new trend has begun. Unlike the Channel Breakout IntraBar, Channel Breakout on Close triggers a buy/sell if the bar's close penetrates the channel.

Usage:

In common usage, a breakout of the upper channel signifies the beginning of a strong upward move while a breakout of the lower channel signifies the beginning of a strong downward move.

The system goes long whenever closes above the highest high of the preceding length bars. It goes long whenever closes above the highest high of the preceding length bars.

System: ChannelBrkWeighted

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

Length10		Number of bars for price channel
----------	--	----------------------------------

Description:

This signal is based on the logic that when the market breaks the upper/lower channel of previous prices on a weighted close basis, a new trend has begun. If the lower channel border is broken, then a sell order will be generated. If the upper channel border is broken, then a buy order will be generated.

Usage:

In common usage, a breakout of the upper channel signifies the beginning of a strong upward move while a breakout of the lower channel signifies the beginning of a strong downward move.

System: ConsecutiveCloses

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

LEConsec	3	Number of consecutive higher closes required for a long entry.
----------	---	--

SEConsec	3	Number of consecutive lower closes required for a short entry.
----------	---	--

Description:

This is a simple technique which generates an order on the close whenever extreme strength or weakness is present in the market. The theory here is that, if the market has closed in one direction consecutively for x number of bars, a major move is developing.

Usage:

Can be used as is, in which case it buys on strength and sells on weakness. Or its signals can be reversed, so that it buys on weakness and sells on strength.

System: Divergence

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

OSC	RSI(Close, 14)	Second input in Bullish Divergence
-----	----------------	------------------------------------

Description:

The Divergence function is used to generate entry orders whenever a divergence occurs between the close and a 14 bar RSI. By typing in different inputs, you can easily modify this signal so that it uses a different oscillator.

Usage:

A bullish divergence occurs when the following occurs:

1. Price makes a swing low, rallies, and then makes another swing low whose price is lower than the prior swing low.
2. Oscillator makes a swing low, rallies, and then makes another swing low whose level is not as low as the prior swing low.

A bearish divergence occurs when the following occurs:

1. Price makes a swing high, dips, and then makes another swing high whose price is higher than the prior swing high.
2. Oscillator makes a swing high, dips, and then makes another swing high whose level is not as high as the prior swing high.

System: KeyReversalMajor

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

OSC	RSI(Close, 14)	Second input in Bullish Divergence
-----	----------------	------------------------------------

Description:

The Divergence function is used to generate entry orders whenever a divergence occurs between the close and a 14 bar RSI. By typing in different inputs, you can easily modify this signal so that it uses a different oscillator.

Usage:

A bullish divergence occurs when the following occurs:

1. Price makes a swing low, rallies, and then makes another swing low whose price is lower than the prior swing low.
2. Oscillator makes a swing low, rallies, and then makes another swing low whose level is not as low as the prior swing low.

A bearish divergence occurs when the following occurs:

1. Price makes a swing high, dips, and then makes another swing high whose price is higher than the prior swing high.
2. Oscillator makes a swing high, dips, and then makes another swing high whose level is not as high as the prior swing high.

System: MACD

Category:

System to Include

Inputs:

None

Description:

A key reversal is a commonly followed chart pattern which identifies shifts in market momentum. A buy signal is generated when the current bar's low is below the previous bar's low and the current bar's close is above the previous bar's close. A sell signal is generated when the inverse occurs.

The Key Reversal Major system is very similar to the standard Key Reversal, except that it is much more selective. In order for it to generate a signal, the high or low must be an eight day extreme.

Usage:

Even though this system is more selective than a standard Key Reversal, it still generates quite a few false signals. Consider using it as a filter with one or more other entry signals.

System: MovAvg(3)Crossover

Category:

System to Include

Function:

MACD(PRICE,FASTMA,SLOWMA)

Inputs:

PRICE specifies some price of the asset of interest

FASTMA number of trailing bars to consider for the fast average

SLOWMA number of trailing bars to consider for the slow average

Description & Usage:

The function returns the MACD value for the current bar, which is the difference between a fast and slow moving average based on the same PRICE. When the function is used in a study or system, PRICE, FASTMA, and SLOWMA can either be hard coded or can be replaced by inputs.

The argument PRICE, is usually hard coded with some bar attribute such as Close, Open, Low, Low, and Volume or is replaced with a numeric series type input. However, it doesn't have to be as simple as this. The PRICE argument, can be replaced with a valid Easy Language expression. For example, Close + Open, or Average(RSI(Close,14),14).

FASTMA and SLOWMA refer to the number of bars used in the moving averages. Therefore, if hard coded, the value used to replace these two arguments should be a whole number and can not change on a bar-by-bar basis. FASTMA is supposed to be less than SLOWMA. If the specified length of FASTMA is greater than that of the SLOWMA, the oscillator will invert.

During rising markets the fast period moving average will rise faster than the slow period moving average resulting in a rising differential line or a larger value. The idea is that with a smaller number of bars used in its calculation, the fast period moving average, will more quickly indicate the trend of the most recent data.

When the fast period moving average crosses above or crosses below the slow period moving average, a signal is generated. Once the lines cross, it becomes important to look at the distance between them. The MACD function returns that difference.

When the fast period moving average is above the slow period moving average the MACD is positive and when it is below the slow period moving average the MACD is negative.

System: MovAvgCrossover

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

Length1	4	Length of moving average #1
---------	---	-----------------------------

Length2	9	Length of moving average #2
---------	---	-----------------------------

Length3	18	Length of moving average #3
---------	----	-----------------------------

Description:

Moving averages are understood and used by more traders than any other technical analysis technique. This particular moving average system is based on closing prices. A buy order is generated when the Fast MA crosses from below to above the Slow MA. A sell order is generated when the Slow MA crosses from above to below the Fast MA.

Usage:

A buy order is generated when the fast average and the medium average cross above the slow average. A sell order is generated when the fast average and the medium average go below the slow average.

System: Parabolic

Category:

System to Include

Inputs:

Name	Description
------	-------------

AF	specifies the Acceleration Factor
----	-----------------------------------

Description & Usage:

The Parabolic system returns the current Parabolic SAR value, and, as described in technical analyst Welles Wilders book, New Concepts In Technical Trading Systems, is a complete stop-setting entry and exit trading system. The system is designed to allow more leeway or tolerance for contra-trend price fluctuation early in a new trade, then to progressively tighten a protective trailing stop order as the trend matures.

The concept draws on the idea that time is an enemy, and unless a trade can continue to generate more profits over time it should be liquidated. Thus, the Parabolic Time/Price System rides the trend until the SAR price is penetrated. Then the existing position is closed out and the reverse position is opened.

The expression parabolic derives from the shape of the curve the stops create as they appear on the chart (this can be seen when the Parabolic indicator is applied to a data series on a chart window).

To calculate the function, we must first find an extreme reference point. On the long side this price is usually the lowest price recorded during the previous closed short position. On the short side this extreme price is usually the highest price recorded during the previous closed out long position.

In practice, however, you won't have an extreme reference point for the very first trade as there are no previous trades. To account for this, the function uses, as a default, either the high or low of the previous bar before the first trade.

For long-side sells and short-side buy, on the second day and thereafter, the SAR is adjusted as follows:

$$\text{SAR}_b = \text{SAR}_p + \text{AF}(\text{H} - \text{SAR}_p)$$
$$\text{SAR}_s = \text{SAR}_p + \text{AF}(\text{L} - \text{SAR}_p)$$

where ...

SAR_b is the long-side sell stop price at which you sell and sell short.

SAR_s is the short-side buy stop at which you buy and go long.

SAR_p is the previous bars SAR

AF is an acceleration factor that begins at .02 for the bar after the initial SAR buy stop order opens the long trade and is increased by .02 each period that price rises to the highest level (H) since the current long trade was opened. For periods when price does not set a new high within the current long trade time duration, AF is left unchanged from its previous periods level.

H is a new highest price since the current long trade was opened on a stop buy order.

L is a new lowest price since the current short trade was opened on a stop sell order (Colby and Meyers, page 379).

The value returned by the Parabolic function is the SAR value described above.

System: PercentROscillator

Category:

System to Include

Inputs:

Name Default

AvgLen 30

PrctRLen 10

BuyLvl 20

SelLvl 80

Description:

When used by themselves, overbought/oversold oscillators are notorious for generating many false entry signals. So we've made this Percent R system a bit more selective by adding a second entry condition. It generates a buy only if the 30 bar moving average moves up while the Percent R is below 20, and it generates a sell order when the 30 bar moving average moves down while the Percent R is above 80.

Usage:

Attempts to generate buy/sell entry orders when market is about to rectify an overbought/oversold condition.

System: RSIOscillator

Category:

System to Include

Inputs:

Name Default Description

RSILength 10 Length of RSI

BuyZone 30 Identify Oversold region

SellZone 70 Identify Overbought region

Description:

The Relative Strength Index (RSI) is an overbought/oversold oscillator that returns a value from 0 to 100. The higher the number, the more overbought the market; the lower the

number, the more oversold the market. First developed by Welles Wilder in the late 70's, the RSI has become one of the most popular technical indicators in use today.

Usage:

The most typical method of employing the RSI is to sell when its value goes above 70 and buy when its value falls below 30.

System: StochasticCrossover

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

Length10		Length associated with the SlowK and SlowD
----------	--	--

Description:

Stochastic theory is based on the assumption that as prices increase, closing prices tend to accumulate ever closer to the highs for the period. As prices decrease, closing prices tend to accumulate ever closer to the lows for the period. Two values are calculated when using Stochastics: %D and %K. It is the divergence of these two values which generates trading orders.

Usage:

When %K is greater than %D, a buy signal is generated. When %D crosses from below to above %K, a sell signal is generated.

System: WeightedAvgCrossover

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

Length1	4	Length of moving average #1
---------	---	-----------------------------

Length2	9	Length of moving average #2
---------	---	-----------------------------

Description:

Weighted Moving Averages are based on the reasoning that what happened two days ago is less important than what happened yesterday. Consequently, instead of giving each day equal importance, a weighted moving average gives each successive day slightly more importance than its predecessor.

Usage:

When the fast weighted average crosses above the slow weighted average, a buy order is generated. When the slow moving average crosses above the slow weighted average, a sell order is generated.

System: XaverageCrossover

Category:

System to Include

Inputs:

Name	Default	Description
------	---------	-------------

Length1	4	Length of moving average #1
---------	---	-----------------------------

Length2	9	Length of moving average #2
---------	---	-----------------------------

Description:

Although the name may seem menacing, an Exponential Moving Average is simply another form of a weighted moving average. Through geometric progression each older price is given less and less importance.

Usage:

When the fast exponential average crosses above the slow exponential average, a buy order is generated. When the slow exponential average crosses above the slow exponential average, a sell order is generated.